OPTIMIZACIÓN DE TRANSFORMACIONES LINEALES DE DATOS MEDIANTE BUSQUEDA LOCAL

INGENIERIA INFORMATICA

AUTOR: FRANCISCO GODOY MUÑOZ-TORRERO

TUTOR: JOSE MARIA VALLS FERRAN
CO-DIRECTOR: RICARDO ALER MUR

Contenidos

Ilustraciones

Tablas

Capítulo 1

Introducción

El objetivo del presente documento es ofrecer al lector una visión global acerca del proyecto realizado. En este primer capítulo se ofrece un resumen del proyecto desarrollado, y descrito en los sucesivos capítulos, así como una declaración de objetivos que se persiguen con la realización del proyecto.

Además, se describe la estructura del documento ofreciendo una breve explicación del contenido de los diferentes capítulos.

1.1 Resumen del proyecto

La clasificación es una de las tareas del aprendizaje automático. El objetivo es separar distintas regiones del espacio de datos que pertenecen a distintas clases o categorías. En ocasiones se puede mejorar la discriminación de las clases transformando el espacio de datos. Estas transformaciones pueden ser de muchos tipos, siendo las más sencillas las transformaciones lineales, las cuales pueden representarse por medio de matrices. Por ejemplo **x' = m*x** representaría la transformación del dato **x** (representado con un vector) en el dato **x'**, mediante la matriz **m**. El objetivo principal de este proyecto es encontrar la transformación óptima desde el punto de vista de la clasificación. O lo que es lo mismo, la matriz **m** que optimiza la precisión de un algoritmo de clasificación concreto en un dominio determinado.

En el presente proyecto se ha realizado la implementación de un algoritmo de búsqueda local, capaz de encontrar matrices que mejoren la clasificación de los datos con un algoritmo de clasificación, que en este caso es el algoritmo K-NN (vecinos más cercanos).

La búsqueda local es capaz de optimizar tres tipos de matrices (o individuos).

1. **Completa**. Matriz que puede poseer cualquier valor en sus elementos.
2. **Diagonal**. Matriz que puede poseer cualquier valor en los elementos de su diagonal. El resto de sus elementos serán 0.
3. **Simétrica**. Los elementos de esta matriz pueden poseer cualquier valor, siempre respetando que la matriz debe ser simétrica.

Paralelamente se ha incluido un individuo que representa la matriz identidad. Este individuo se utilizará para comparar los resultados obtenidos con los individuos que el algoritmo ha ido evolucionando respecto a la matriz identidad, que se considera el caso base. La matriz identidad equivale a no hacer ninguna transformación.

Además, se han implementado dos variaciones de la búsqueda local, que se han denominado normal y N hijos, que serán descritas en el capítulo 3.

1.2 Objetivo

En el campo del aprendizaje automático es importante contar con atributos que separen correctamente los datos de diferentes clases. Por ello, con el desarrollo del proyecto se persiguen los siguientes objetivos:

- Implementar un algoritmo de búsqueda local que permitan evolucionar matrices de transformación para incrementar el porcentaje de aciertos en problemas de clasificación. Se usará como algoritmo base el de los k-vecinos (K-NN).
- Validar el sistema desarrollado con dominios tanto reales como sintéticos

1.3 estructura del documento

Este documento, que describe el proyecto de búsqueda local desarrollado, se divide en los siguientes capítulos.

- **Capítulo 2: Contexto**. Este capítulo describe el contexto en el que se ha desarrollado el proyecto. En él se describen algunas técnicas evolutivas a modo de ejemplo. También se describe el algoritmo K-NN utilizado en la fase de entrenamiento del algoritmo desarrollado.
- **Capítulo 3: Sistema Desarrollado**. En este capítulo se describe el sistema desarrollado, describiendo cada uno de los pasos que componen el algoritmo final.
- **Capítulo 4: Experimentación**. El capítulo de experimentación muestra los resultados obtenidos con el algoritmo desarrollado y los compara con la matriz identidad. Para realizar este proceso comparativo se han utilizado diferentes dominios de datos.
- **Capítulo 5: Conclusiones**. Este último capítulo recoge las conclusiones obtenidas de acuerdo al proyecto realizado y a los resultados obtenidos en la fase de experimentación. Además se proponen líneas futuras de investigación.

Capítulo 2

Contexto

2.1 Búsqueda Local

A lo largo de esta sección se describirá en qué consiste el método de Búsqueda Local y se indicarán los principales métodos de búsqueda local utilizados para la resolución de problemas.

2.1.1 Introducción

Se define un método de *Búsqueda Local* como un método algorítmico de búsqueda en un espacio de soluciones candidatas que comienza desde un candidato a solución y desde éste va iterando basándose en sus vecinos directos o información local, hasta alcanzar la solución final o la condición de satisfacción.

En la práctica los métodos que implementan la búsqueda local han ido creciendo a lo largo de las últimas décadas. Los métodos incluyen *heurísticas* para dirigir la búsqueda y permitir avanzar hasta alcanzar la solución, es decir, tienen la capacidad para realizar de forma inmediata innovaciones positivas para sus fines.

El principal problema que presentan los algoritmos con mejora iterativa es que pueden estancarse fácilmente en un mínimo local de la solución del problema. Para solucionar este problema se *juega* con la complejidad del vecindario de los candidatos a la solución.

Algunos de los métodos que se pueden encontrar dentro de las búsquedas locales son los siguientes:

- Aleación simulada (Simulated annealing).
- Búsqueda Tabú (Tabu search).
- Algoritmos evolutivos (Evolutive algorithms).

Estos métodos que se pueden utilizar para la implementación de la búsqueda local se detallarán en los siguientes apartados.

2.1.2 Aleación simulada (Annealing simulated)

El algoritmo de aleación simulada pertenece a la clase de algoritmos de búsqueda local conocidos como algoritmos con umbral.

El nombre del algoritmo proviene del proceso de recocido del acero, técnica que se basa en calentar y posteriormente enfriar. Los átomos mediante el calor salen de sus posiciones iníciales y se mueven aleatoriamente. Al enfriarse tratan de encontrar configuraciones con menor energía.

El objetivo de este método es resolver el problema de la optimización y obtener el estado de mínima energía.

Inicialmente se parte de un estado solución generado de forma aleatoria. *Annealing* considera los vecinos del estado actual y probabilísticamente cambia de estado. Este proceso se repetirá hasta alcanzar la solución. El paso de un estado a otro se realiza a través de una función de energía.

Simulated Annealing comienza realizando una amplia exploración, para evitar que el resultado dependa del estado inicial y evitar encontrarse con un mínimo/máximo local.

La probabilidad de pasar a estado de mayor energía es:

$$p' = e^{-\Delta E/kT}$$

Imagen 1: Cálculo de probabilidad en Simulated Annealing [1]

Donde ΔE es el cambio positivo de energía, T la temperatura y k se corresponde con la constante de Boltzmann.

El *annealing* puede usar variedad de predicados de terminación, por lo que a menudo utiliza una condición de terminación basada en un rango de aceptación. De esta forma el proceso de búsqueda se detendrá cuando la solución esté dentro del rango de aceptación o cuando el algoritmo no mejore el candidato solución en el número de pasos establecidos para la búsqueda.

A continuación se describe el pseudocódigo del algoritmo:

```
Algoritmo SA (Est_inicial, Objetivo, Estrategia_aleanning)

Evaluar Est_inicial
Si es una meta →devolver Est_inicial y fin
Si no, Est_actual = Est_inicial
        Est_mejor = Est_inicial
Inicializar T con Estrategia_aleanning
Mientras se busca solución o hasta que no se pueda continuar aplicando operadores
        Seleccionar operador a aplicar al Est_acutal
        Evaluar Est_actual
        Calcular ΔE = valor(Est_actual)-valor(Est_nuevo)
        Si Est_nuevo es meta → devolver Est_nuevo y fin
        Si no, Si Est_nuevo es mejor que Est_actual entonces Est_actual = Est_nuevo
                                                              Est_mejor = Est_nuevo
                Si no, Est_actual = Est_nuevo con probabilidad p'
        Revisar T de acuerdo a Estrategia_aleanning
Devolver Est_mejor
```

Imagen 2: Pseudocódigo Simulated Annealing

Las operaciones acerca de cuándo disminuir el valor de T, o el criterio para bajar el valor de T se contemplan en la estrategia de *Annealing*. Además, el valor inicial de T también se establece en la estrategia.

2.1.3 Búsqueda Tabú

El origen de esta técnica de búsqueda local se remonta a los años 1970. Aunque fue formalmente presentada por Glover en 1986.

Este tipo de método de búsqueda fue diseñado para mejorar la eficiencia del proceso de exploración. Al igual que otros métodos utiliza los registros de información local como valor actual de la función, pero además aporta información que se relaciona con el proceso de exploración. Este proceso provoca que se restrinjan los posibles vecinos sobre los que avanzar, haciendo que la vecindad sea dinámica.

Se conoce como una técnica metaheurística, debido a que posee técnicas heurísticas que se utilizan para dirigir la búsqueda.

A diferencia de otras técnicas de búsqueda local, el método de búsqueda tabú únicamente emplea un individuo (son múltiples las técnicas que emplean poblaciones con un amplio número de individuos). Pero además, también se diferencia de otras técnicas por el uso de memoria.

El uso de una memoria adaptativa permite al método de búsqueda tabú controlar el proceso de búsqueda que permita encontrar el óptimo resultado para resolver el problema. La forma de gestionar esta memoria es manteniendo un histórico de los estados (con todos los atributos) visitados. También se pueden almacenar las el número de ocurrencias de atributos en las diferentes soluciones alcanzadas. Esta estrategia puede ser usada para dirigir movimientos, proporcionando un mayor peso a los atributos que más se repiten.

Los últimos movimientos o iteraciones realizadas se consideran tabús para evitar que en las futuras iteraciones se vuelvan a repetir y el algoritmo entre en un ciclo.

A continuación se describe el pseudocódigo del algoritmo:

Algoritmo TS ()

1. Seleccionar solución inicial *i*
2. Insertar *i* en la lista tabú
3. Elegir mejor *j* en el vecindario
4. Si f(j) >= f(j) → fin
5. Si no *i* = *j* entonces ir al paso 2

Imagen 3: Pseudocódigo Búsqueda Tabú

La condición de parada de este método será establecida por el usuario, pudiendo elegir entre varias:

- Número de iteraciones fijo.
- Alcanzar una solución mejor que un valor fijado.
- No mejorar una solución tras un número de iteraciones fijado.
- Todos los vecinos están incluidos en la memoria.

El éxito de la *Búsqueda Tabú* dependerá del correcto modelado del problema. Es importante que la vecindad esté correctamente definida, así como la función objetivo.

2.1.4 Algoritmos evolutivos

Debido a la dificultad de resolución de algunos problemas de optimización por medio de técnicas tradicionales, se emplean algoritmos evolutivos inspirados en la naturaleza para ello.

Este tipo de algoritmos comenzaron a ser utilizados en 1960. En un trabajo realizado por Bremmermann en 1966 escribió:

El propósito de este trabajo es estudiar los efectos de la mutación, el cruce, la selección y la evolución de los genotipos en los casos en los que la función de fitness no es lineal. En vista a las dificultades matemáticas que envuelven a la resolución de los problemas de optimización, los equipos de experimentación han sido utilizados en combinación con análisis teóricos… En nuevas series de experimentos nosotros encontramos que los esquemas evolutivos convergían mucho mejor, pero sin equivalencias biológicas conocidas.

Los algoritmos evolutivos son métodos de búsquedas con un gran poder de optimización inspirados en la naturaleza. Se basan principalmente en la selección natural y la reproducción.

Los algoritmos evolutivos comienzan con la generación aleatoria de una población de individuos. Estos individuos son representaciones de posibles soluciones.

A cada uno de los individuos de la población o generación se le asignará un valor conocido como *fitness*. El valor de fitness determinará cómo de buena o mala es la solución representada por el individuo.

Tras asignar el valor de fitness a todos los individuos de la generación, se procede a transformar la generación. Para esta tarea se utilizan los métodos de:

- Selección de individuos. La selección elegirá a los mejores individuos de la generación para proseguir con la especie.
- Reproducción. Los individuos se reproducen o se cruzan. Se obtienen varios hijos a partir de unos progenitores.
- Mutación de individuos. Algunos alelos del cromosoma o individuo pueden sufrir alteraciones o mutaciones, variando así su valor.

Finalmente, y con los nuevos individuos generados se crea una nueva generación que sustituye o reemplaza a la que había hasta este momento.

Este proceso se ejecuta un número de generaciones establecidas desde el comienzo. Cuando el número de generaciones alcanza el límite establecido, se obtendrá el mejor individuo generado a lo largo del proceso evolutivo que representará la mejor solución para el problema de optimización.

El siguiente diagrama muestra la secuencia de ejecución de un algoritmo evolutivo.

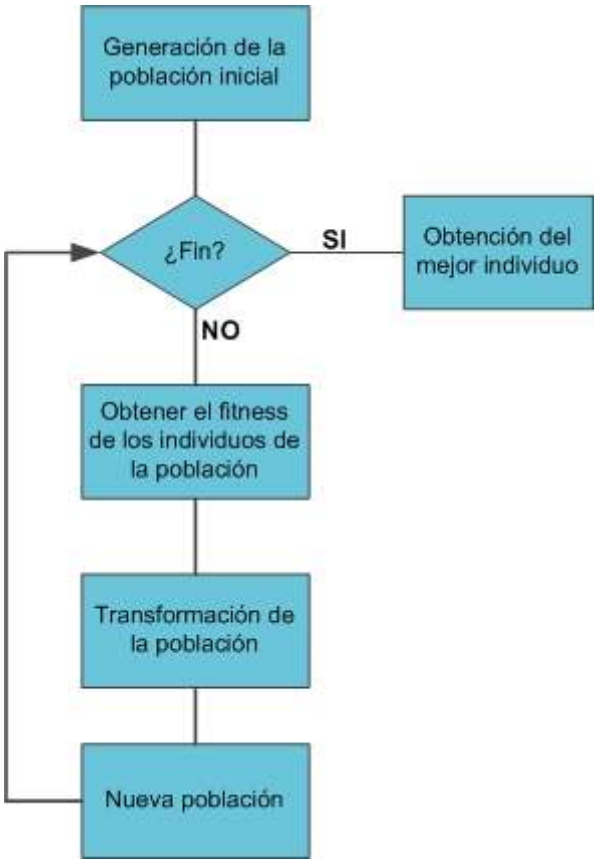


Imagen 4: Flujo de ejecución de los Algoritmos Evolutivos

Algunos métodos evolutivos que siguen este esquema son:

- Estrategias evolutivas.
- Programación evolutiva.
- Algoritmos genéticos.
- Programación genética.

Los algoritmos genéticos son una de las técnicas evolutivas más utilizadas. La primera teoría formulada acerca de ellos fue proclamada por Holland en 1975. Dicha teoría asumía, a alto nivel, que los algoritmos genéticos trabajaban descubriendo y combinando buenos "bloques construidos" [4].

2.2 Algoritmo K-NN

El nombre completo del algoritmo es k-Nearest Neighbor, es decir los k vecinos más cercanos. En los siguientes apartados se dará una amplia visión de este algoritmo.

2.2.1 Introducción

El algoritmo k-Nearest Neighbor, a partir de ahora K-NN, asume que todas las instancias se encuentran en el espacio de *n* dimensiones (R^n).

Requiere que el valor que representa la clase a la que pertenece cada instancia sea un valor discreto o un valor real.

K-NN es un algoritmo del tipo *Instance-Based Learning*. Este tipo de algoritmos almacena los datos o instancias utilizadas durante la fase de almacenamiento (conjunto de datos de entrenamientos) para realizar una estimación o clasificación basada en dicho conjunto.

El objetivo del algoritmo es clasificar una nueva instancia nunca vista definiendo para ello la clase a la que pertenecerá esta instancia.

2.2.2 Descripción del algoritmo

El algoritmo K-NN está basado en instancias, tal y como se mencionó con anterioridad. Se considera una instancia como un conjunto de datos que reflejan el valor de diferentes atributos. Además, las instancias almacenadas dispondrán de la clase a la que pertenecen.

La nomenclatura utilizada para representar instancias a lo largo del proyecto será la descrita a continuación.

Se considerará una instancia *x* como un conjunto de valores a_i que representan los valores para cada uno de los atributos que componen la instancia.

$$(a_1(x), a_2(x), a_3(x), \dots, a_n(x))$$

Imagen 5: Representación de una instancia

La instancia x se describe como una sucesión de valores. Por ejemplo, valor del atributo 1 de la instancia x, valor del atributo 2 de a instancia x, y así con todos los atributos que la componen.

Una vez descritos estas nociones básicas para poder seguir la descripción del algoritmo, se procede a detallar el mismo.

El algoritmo K-NN dispone de un conjunto de instancias almacenadas, sobre las que se realizará la clasificación de nuevas instancias.

El método empleado para clasificar la nueva instancia es mediante el uso de la distancia Euclidea.

$$d(A,B) \equiv \sqrt{\sum_{i=1}^{n} (a_i(A) - a_i(B))^2}$$

Imagen 6: Fórmula de la Distancia Euclidea [3]

La distancia Euclidea entre las instancias A y B es la raíz cuadrada del sumatorio de las diferencias entre los atributos de cada instancia al cuadrado.

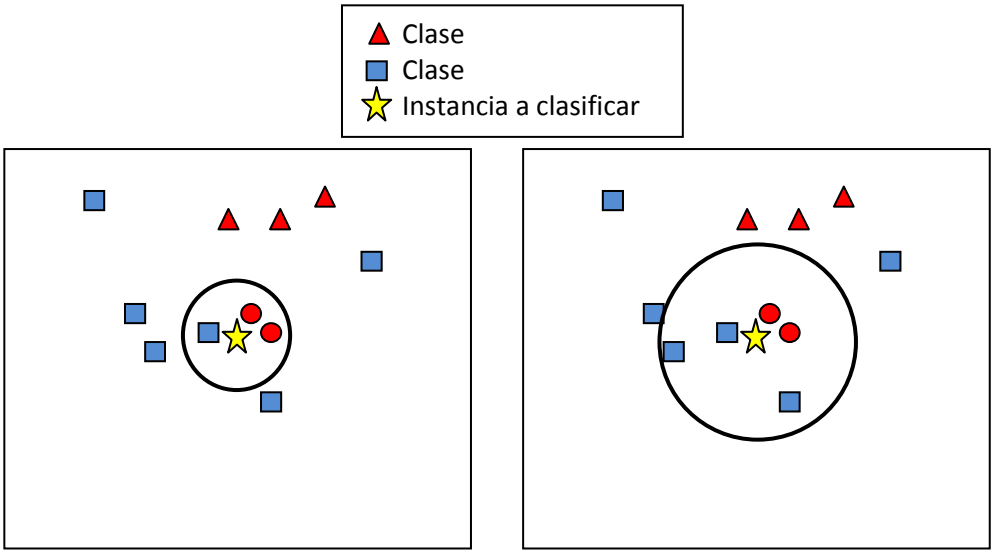
Se calculará la distancia euclidea de la nueva instancia respecto a cada una de todas las instancias con las que ya cuenta el algoritmo. Es en este punto, donde entra en funcionamiento el valor *k* que introduce al algoritmo.

La clase resultante de la instancia a clasificar será calculada respecto a las *k* instancias que minimicen la distancia euclidea o *k* vecinos más cercanos. Por ello finalmente la clase será la que más veces se repita en estos *k* vecinos más cercanos (en caso de igualdad, será aleatoriamente una de ella).

2.2.3 Ejemplo de uso

A continuación se muestra un ejemplo de gráfico que recoge el funcionamiento del algoritmo:

El ejemplo descrito pretende clasificar una nueva instancia mediante el uso del algoritmo K-NN. Para ello utilizará las instancias en la que basa su aprendizaje. Se proponen dos situaciones, variando el valor de *k*.



Este ejemplo utiliza un valor *k*=3.

La clase que se le asignaría a la nueva instancia será ▲ , ya que de las 3 menores distancias (señaladas con el círculo), dos pertenecen a instancias de la clase triángulo y solamente una a la clase cuadrado.

Este ejemplo utiliza un valor *k*=5.

La clase que se le asignaría a la nueva instancia será ■ , ya que de las 5 menores distancias (señaladas con el círculo), dos pertenecen a instancias de la clase triángulo y tres a la clase cuadrado.

Imagen 7: Ejemplo Algoritmo k-NN

2.3 Distancias y proyecciones

Existen diferentes formas o funciones para determinar la distancia entre dos datos dados. En este apartado se describirán tres funciones de distancias:

- Distancia euclidea no ponderada
- Distancia euclida ponderada
- Distancia euclidea generalizada

2.3.1 Distancia euclidea no ponderada

La distancia euclidea es la que normalmente se utiliza en el cálculo de distancias, y por ello es la más conocida y usada.

Su función es:

$$d(A,B) \equiv \sqrt{\sum_{i=1}^{n}(A_i - B_i)^2} = \sqrt{(A - B)^T (A - B)}$$

Imagen 8: Fórmula de la distancia Euclidea no ponderada [6]

Para ilustrar el funcionamiento de la función de distancia euclidea, a continuación se presenta gráficamente un ejemplo.

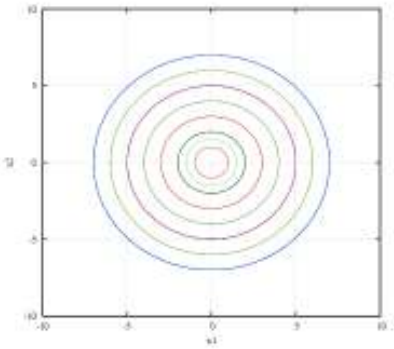


Imagen 9: Ejemplo de función de distancia euclidea no ponderada

En el ejemplo se puede observar como para esta función la distancia entre dos puntos es el espacio que los separa. Por ello, cualquier punto situado en alguno de los círculos se encuentra a la misma distancia del centro del mismo.

2.3.2 Distancia euclidea ponderada

La distancia euclidea ponderada es una función de distancia basada en la función de distancia euclidea no ponderada.

Su función es:

$$d(A,B) \equiv \sqrt{\sum_{i=1}^{n} (m_i(A_i - B_i))^2} = \sqrt{(A - B)^T M^T M (A - B)}$$

Imagen 10: Fórmula distancia euclidea ponderada

En este caso M será una matriz diagonal, y $m_i=M_{ii}$ es el factor con el que se escala la dimensión i.

El efecto que provoca esta función de distancia es similar a escalar cada una de las dimensiones.

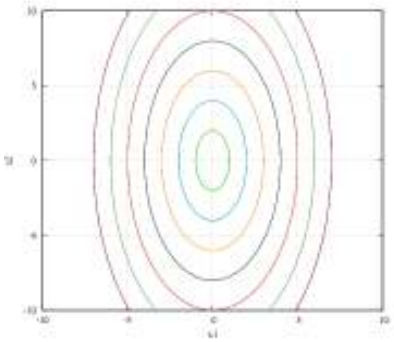


Imagen 11: Ejemplo de función de distancia euclidea ponderada

En el siguiente ejemplo, se puede ver como esta función de distancia recoge que dos puntos se encuentran a la misma distancia si se encuentran en el mismo óvalo. Por ejemplo, cualquier punto situado en el óvalo verde de la imagen se encuentra más "cerca" del centro que cualquier punto del óvalo azul por ejemplo. De esta manera, se ha conseguido acercar ciertos puntos (los que están en la parte excéntrica de las elipses) en comparación con la distancia euclidea. Así se puede ver que simplemente cambiando la matriz M, se modifica la métrica del espacio.

2.3.3 Distancia euclidea generalizada

La distancia euclidea generalizada es una función que complica un paso más con respecto al caso anterior la forma de determinar la distancia entre dos puntos.

Su función es:

$$d_M(A,B) \equiv \sqrt{(A - B)^T M^T M (A - B)}$$

Imagen 12: Fórmula distancia euclidea generalizada

En este caso M será una matriz arbitraria, con cualquier combinación de valores en los elementos que la componen.

Ejemplo gráfico de distancia euclidea ponderada:

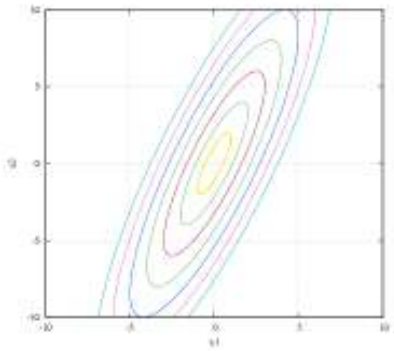


Imagen 13: Ejemplo de función de distancia euclidea generalizada

En el siguiente ejemplo, se puede ver como esta función de distancia recoge que dos puntos se encuentran a la misma distancia si se encuentran en la misma elipse. Por ejemplo, cualquier punto situado en la elipse amarilla de la imagen se encuentra más "cerca" del centro que cualquier punto de la elipse verde por ejemplo.

2.3.4 Proyecciones

La proyección lineal de los datos permite convertir cada dato en un nuevo dato que denominaremos *dato proyectado*. Este nuevo dato obtenido será un dato al cual se le habrán aplicado una serie de factores a cada uno de los atributos o parámetros que componen el dato.

El nuevo dato obtenido, tal y como se ha mencionado con anterioridad, tendrá unos nuevos atributos de acuerdo a la importancia o a los factores que tenga la matriz utilizada.

En el ejemplo que a continuación se presenta, el dato se representa como ***b*** y la matriz que se utilizará para realizar la proyección del dato es ***a***. El resultado de la proyección será el *dato proyectado* ***b'***.

$$\begin{pmatrix} b_1 & b_2 & \dots & b_n \end{pmatrix} * \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ . & . & & . \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} = \begin{pmatrix} b'_1 & b'_2 & \dots & b'_n \end{pmatrix}$$

Imagen 14: Proyección lineal de un dato

Capítulo 3

Sistema Desarrollado

3.1 Introducción

En el presente capítulo se describe el sistema desarrollado sobre búsqueda local. Este algoritmo de búsqueda local será iterativo e irá generando y evaluando posibles candidatos a solución, basándose en los paradigmas de la búsqueda [2]. El algoritmo desarrollado será supervisado, ya que se cuenta con un conjunto de datos de entrenamiento y sus correspondientes clases son conocidas [5].

El sistema ha sido desarrollado en el entorno de programación Eclipse, e implementado en el lenguaje de programación Java.

Java se considera un lenguaje de programación "moderno" que se ha basado en lenguajes como C o C++. Si bien, es equivocado creer que es una evolución de éstos, ya que Java cuenta con características que le otorgan una identidad propia. Java se crea a principios de los años 90 con el objetivo de crear un lenguaje independiente de la plataforma donde se ejecuten los programas. Los creadores de este nuevo lenguaje de programación fueron un grupo de ingenieros de Sun Microsystem.

Para la implementación y ejecución del programa desarrollado se ha utilizado el entorno de desarrollo Eclipse.

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma. Especialmente esta plataforma ha sido utilizada para desarrollar entornos de desarrollo integrados como el IDE de Java, llamado Java Development Toolkit y el compilar que se incluye como parte de Eclipse.

Algunos de los lenguajes de programación para los que se utiliza Eclipse como entorno de desarrollo son:

- Java
- ANSI C
- C++
- Jsp
- sh
- perl
- php

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios [7].

3.2 Diagrama de clases

En este punto, se describirán cada una de las clases implementadas para desarrollar la aplicación final. Una breve descripción de cada uno de los componentes que forman el sistema es incluida a continuación. Dicha descripción será realizada por paquetes:

- **Individuo**. Contiene las clases que representarán durante la ejecución a los diferentes individuos.

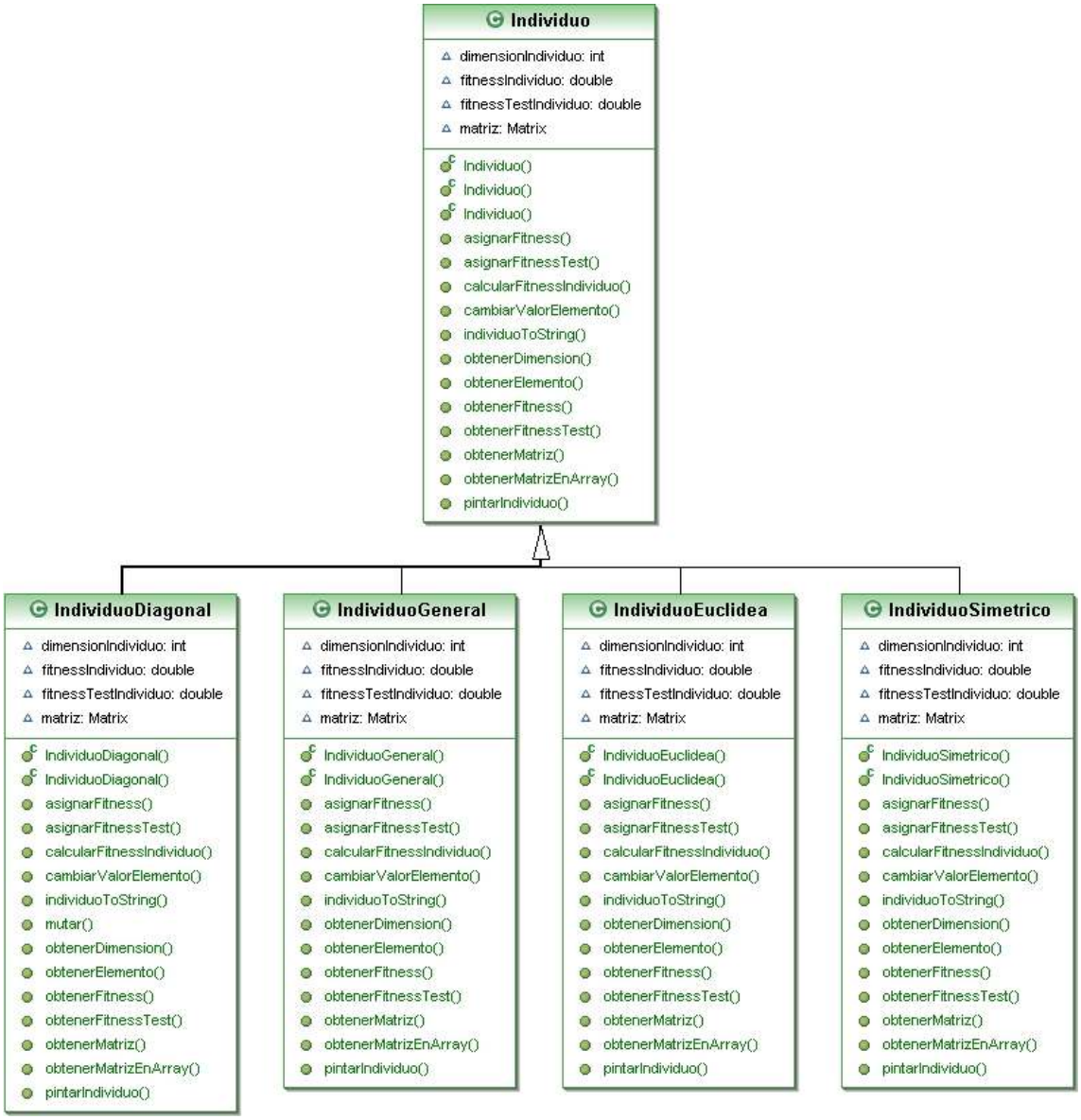


Imagen 15: Paquete Individuo

Observando las clases de este paquete se observa como existe una clase individuo de cual heredan los diferentes tipos de individuos: General, Diagonal, Simetrico y Eucideo (se describirán más adelante en el documento). Los individuos están formados por una matriz, y poseen un fitness de test y otro de entrenamiento.

- **BusquedaLocal**. El paquete contiene los diferentes tipos de búsquedas implementadas, para los diferentes tipos de individuos se encuentran en este paquete.

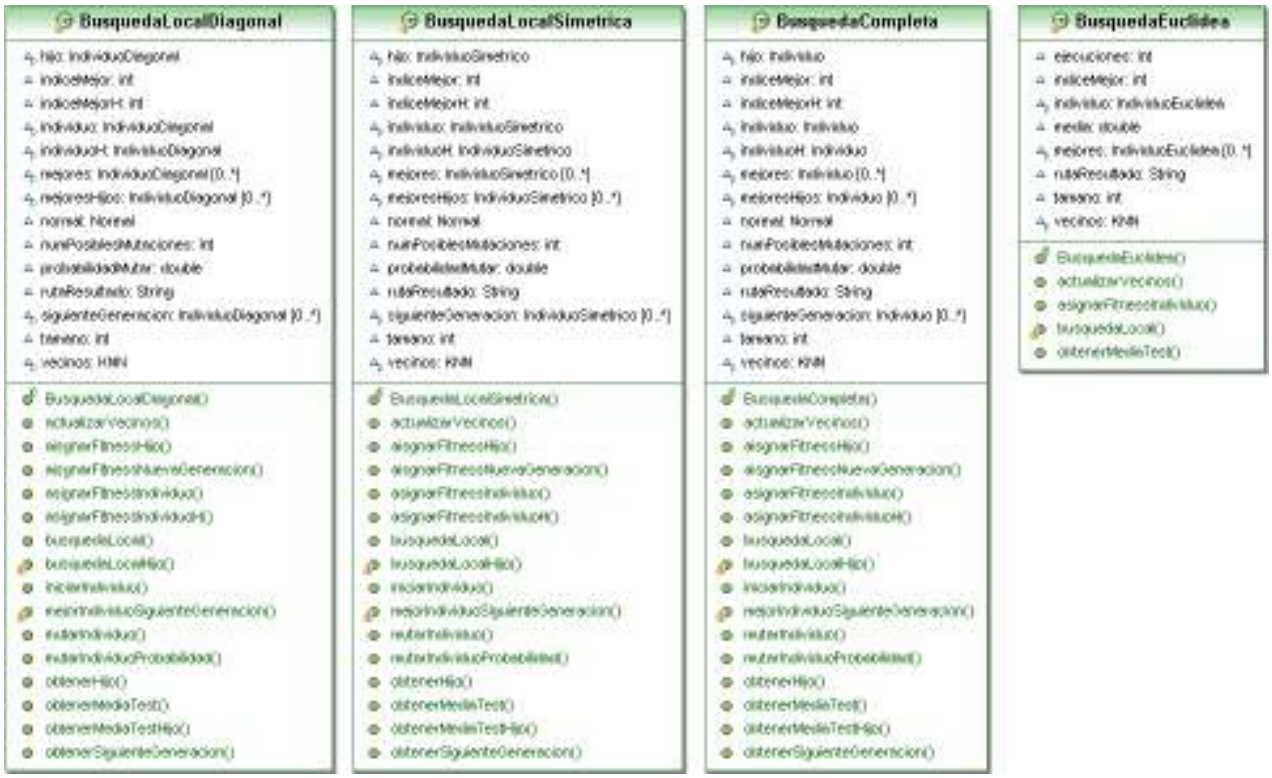


Imagen 16: Paquete BusquedaLocal

Los principales atributos de estas clases son los individuos, cada búsqueda utiliza el tipo de individuo asociado. Además se utiliza un array de individuos descendientes o un individuo hijo, dependiendo de la ejecución seleccionada por el usuario (descrito posteriormente en el capítulo).

Entre los métodos más importantes se encuentran búsquedaLocal() y búsquedaLocalHijo(), ya que realizan la funcionalidad propia de un algoritmo de búsqueda local.

- **KNN**. Este paquete corresponde a la implementación del algoritmo K-NN.

Imagen 17: Paquete KNN

Contiene una clase Dato que es la encargada de transformar los datos leídos desde el fichero a un objeto; y la clase K-NN que corresponda a la implementación del propio algoritmo.

La clase dato contiene un array con los diferentes atributos del dato y una variable que almacena la clase a la que pertenece ese dato.

La clase K-NN dispone de dos conjuntos, entrenamiento y test, los cuales serán utilizados para realizar cada una de las fases requeridas por el algoritmo de búsqueda local. Los métodos más importantes son clasificar() y clasificarTest(). Se encargan de clasificar un dato usando el algoritmo K-NN, en cada una de las fases.

- **Fichero**. Contiene las clases necesarias para realizar las lecturas y escrituras en los ficheros.

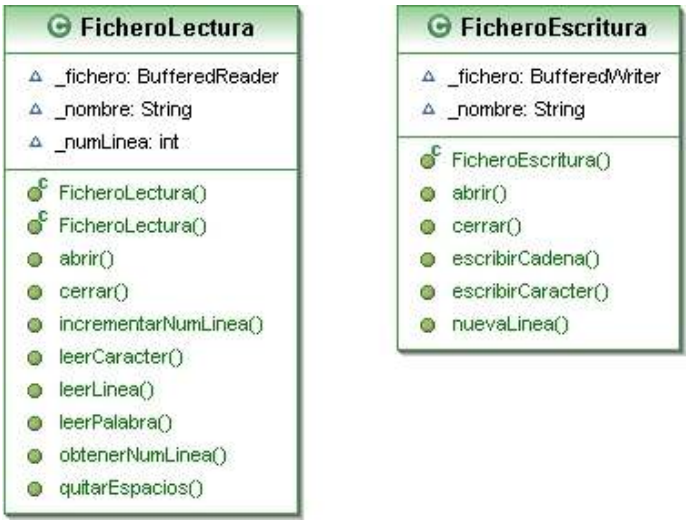
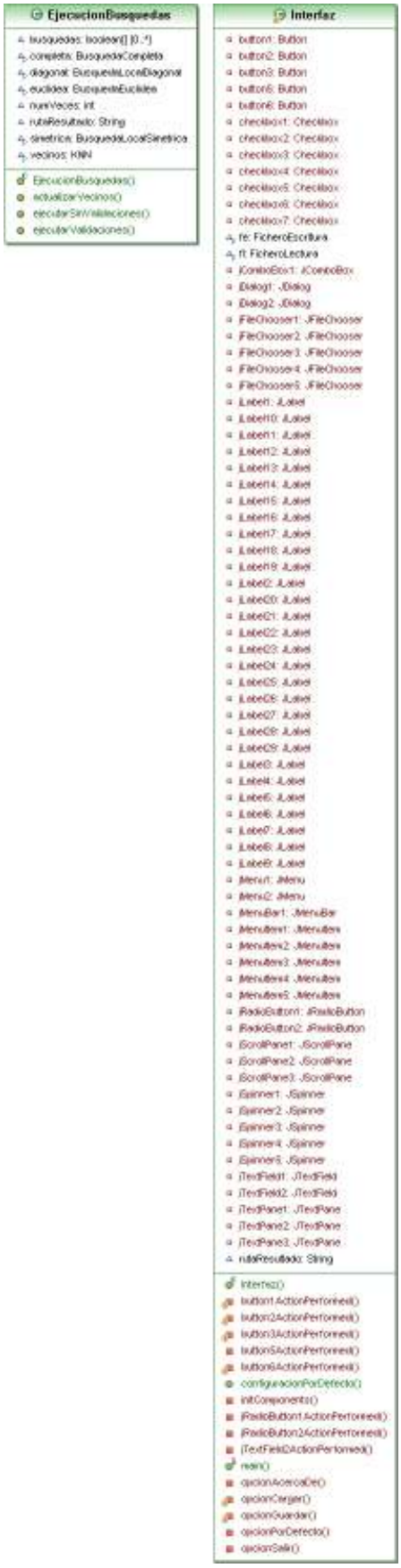


Imagen 18: Paquete Fichero

El uso de la clase FicheroLectura permite al programa desarrollado poder leer el/los fichero/s de datos que el usuario introduzca en la aplicación. Son importantes los métodos abrir(), para poder comenzar a leer datos desde el fichero y cerrar(), para dejar el fichero cerrado una vez se ha finalizado su lectura. Además, el método leerLinea() ha sido muy utilizado para leer las diferentes líneas del fichero de datos, para posteriormente ser tratadas y extraer el objeto de la clase dato que se comentó anteriormente.

La clase FicheroEscritura se ha empleado para poder generar una serie de ficheros con los resultados del entrenamiento y con la evolución del individuo. Al igual que en el caso de la clase FicheroLectura, los métodos abrir() y cerrar() son importantes. Y por último, destacar el método escribirCadena() que ha sido utilizado para ir añadiendo líneas al fichero resultante.

- **Ejecución**. Este paquete es el encargado de poner en funcionamiento la aplicación desarrollada.



Contiene la clase EjecucionBusquedas que se encarga de ejecutar las diferentes búsquedas que el usuario introduzca en la interfaz.

La clase contiene diferentes tipos de búsquedas (completa, diagonal, simetrica y euclidea) y ejecuta las que el usuario haya seleccionado en la interfaz mediante los checkbox correspondientes.

Y también contiene la clase Interfaz, clase que contiene el código que genera la interfaz de usuario con la que se interaccionará con el programa.

La clase interfaz contiene un gran número de atributos, que representas los diferentes elementos que componen la interfaz, tales como: botones, checkbox, paneles, menús, etc.

Dispone un método que restaura en la interfaz la configuración por defecto, y permite además guardar la configuración actual en la ventana y cargar diferentes configuraciones que el usuario haya guardado con anterioridad.

Esta clase Interfaz es el punto de arranque del programa desarrollado.

Imagen 19: Paquete Ejecución

3.3 Descripción del algoritmo

El sistema desarrollado consiste en una búsqueda local basada en la evolución de un individuo. A continuación se muestra un diagrama de flujos que detalla el funcionamiento del algoritmo.

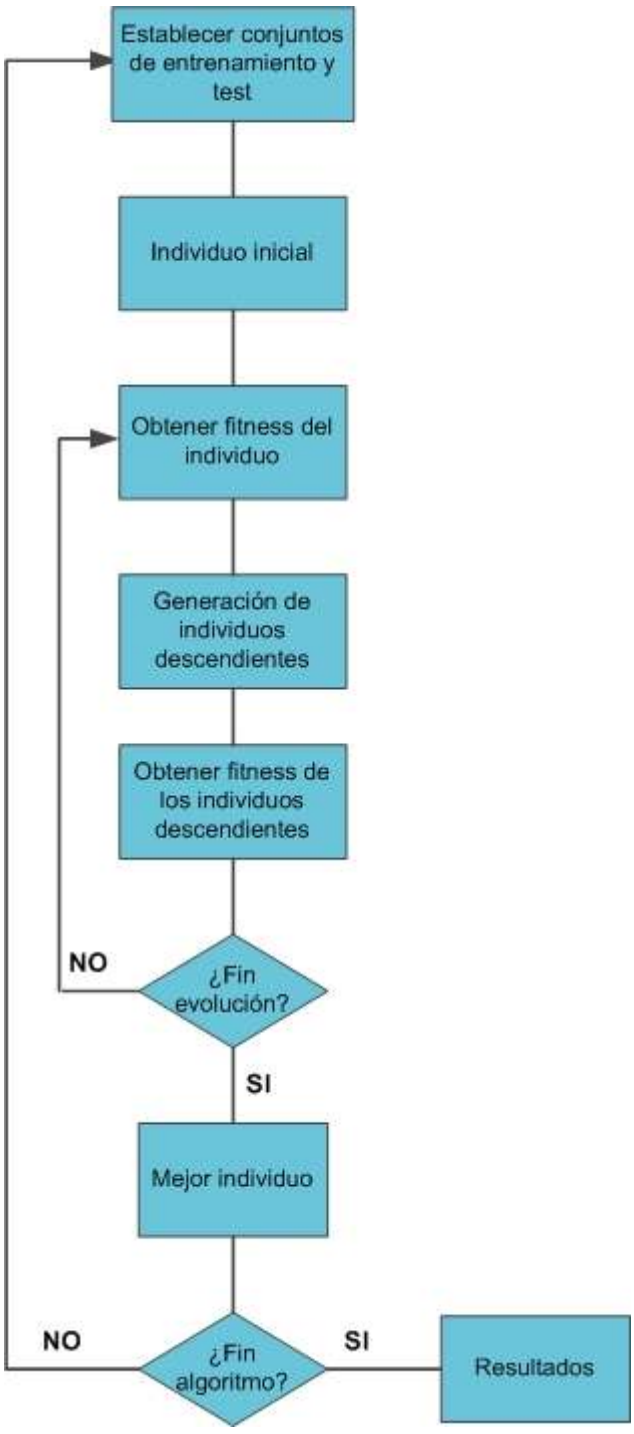


Imagen 20: Flujo del sistema desarrollado

Las fases que componen el algoritmo son las siguientes:

1. **Establecer conjuntos de entrenamiento y test**. El comienzo para la ejecución del programa es establecer los conjuntos de entrenamiento y test que utilizará el algoritmo para evaluar el correcto funcionamiento de la evolución de los individuos.

2. **Generación de un individuo inicial**. En esta fase se crea un individuo que será a partir del cual se desarrollará el algoritmo. Esté individuo se considerará individuo principal.

3. **Obtención del fitness del individuo**. En esta fase se le asignará al individuo un valor de fitness, que en este caso se corresponderá con un porcentaje de instancias acertadas.

4. **Generación de descendientes**. Esta fase es la encargada de generar una serie de individuos descendientes a partir de un método evolutivo y del individuo principal.

5. **Obtención del fitness de los individuos descendientes**. Al igual que se realizó anteriormente con el individuo principal, en esta fase se calculan los fitness de cada uno de los individuos descendientes generados tras la evolución.

6. **Evaluación de la condición de terminación de la evolución**. Cuando se hayan realizado los pasos anteriores, se evaluará la condición de terminación y dependiendo del resultado se realizará una acción u otra. Si el algoritmo ha alcanzo su fin, se obtendrá el mejor individuo conseguido durante el proceso de búsqueda. Por el contrario, si todavía no se satisface la condición de terminación se volverá al paso 2, pero como individuo principal el mejor de los individuos descendientes generados según indique su fitness.

7. **Evaluación de la condición de terminación del algoritmo**. En este punto se evalúa si el programa ha alcanzado su fin.

Los siguientes apartados del capítulo detallan cada una de las fases que componen el algoritmo desarrollado de búsqueda local, detallando los métodos empleados.

3.3.1 Establecer conjuntos de entrenamiento y test

El programa implementado permite realizar una carga de datos a tratar de dos maneras diferentes y excluyentes:

- **Usar validaciones**. Mediante el uso de validaciones el usuario indica un único fichero en el que contiene todos los datos a tratar por el programa. El usuario debe indicar en la pantalla inicial de configuración que se utilizarán validaciones, y además debe indicar el número de validaciones deseadas y la ruta del fichero que contiene los datos. Estas opciones se establecen en las secciones *Parámetros de configuración de K-NN* y *Fichero datos de entrenamiento* de la pantalla de configuración.

  Al emplear esta forma de tratamiento de datos para establecer los conjuntos de entrenamiento y test, se generarán el número de folders indicados y se le asignarán datos leídos desde el fichero indicado de forma aleatoria. Una vez creados los folders se formarán los conjuntos de entrenamiento y test.

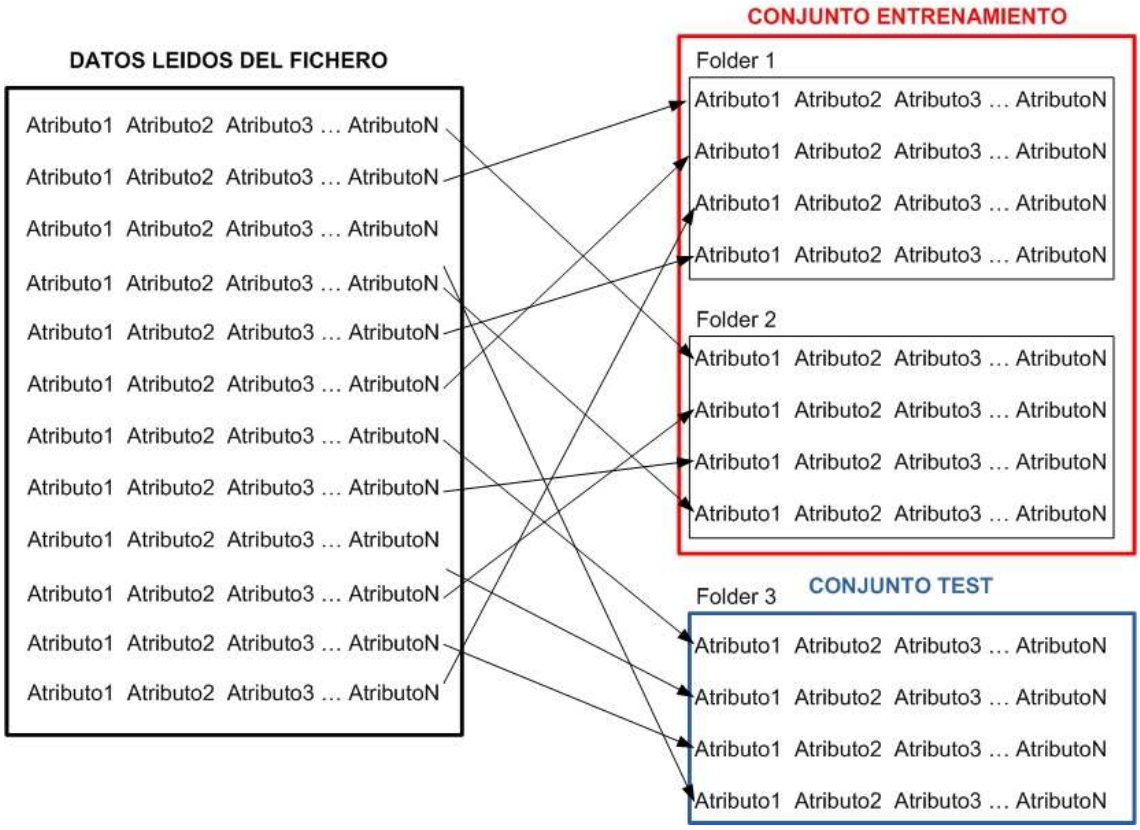


Imagen 21: Conjuntos de entrenamiento y test con validaciones

El resultado de ejecutar el programa con validaciones es la obtención de un número de mejores individuos igual al número de validaciones indicado. Cada uno de estos individuos considerará como conjunto de test un folder distinto.

- **Usar ficheros con datos de entrenamiento y test**. Esta opción no incorpora validaciones. Únicamente establece un conjunto de datos de entrenamiento que contiene los datos leídos desde el fichero de datos de entrenamiento indicado, y un conjunto de datos de test con los datos leídos desde el fichero de datos de test indicado.

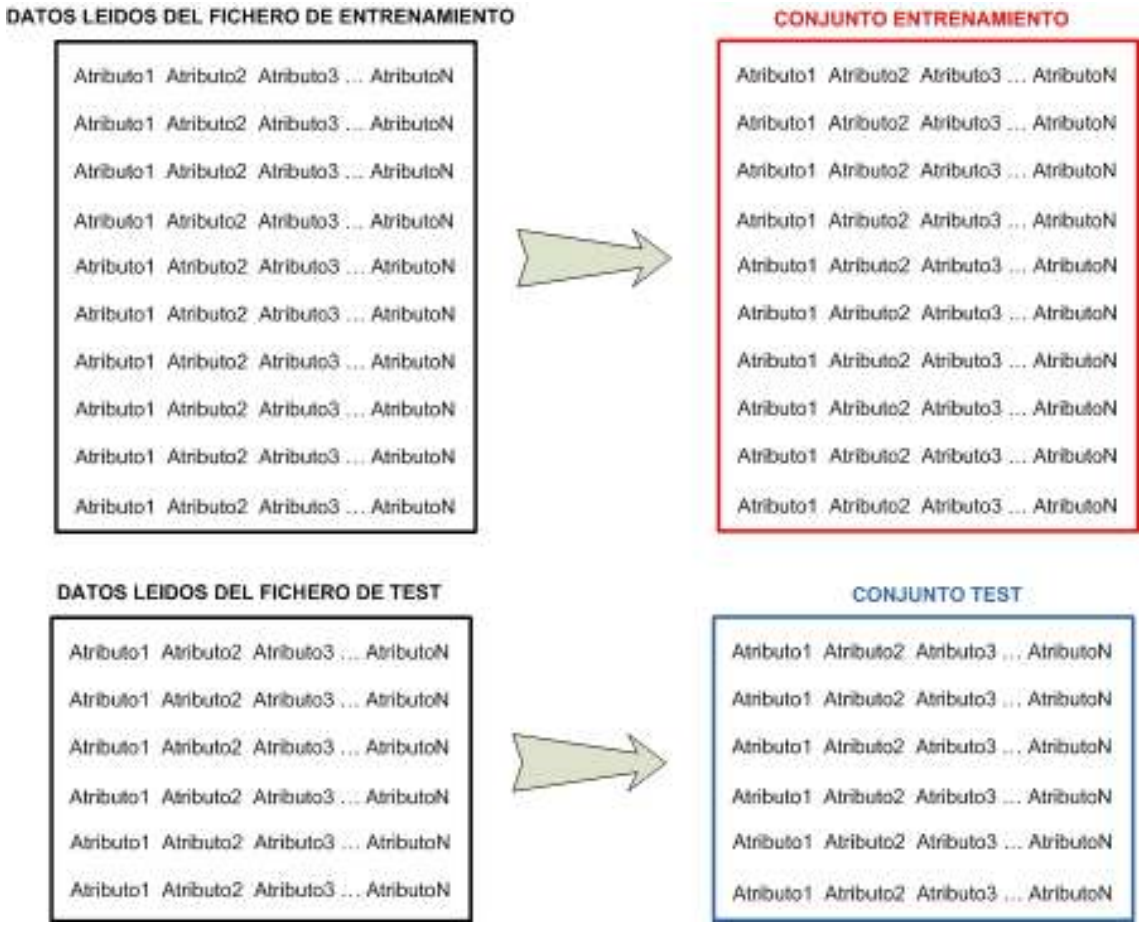


Imagen 22: Conjuntos de entrenamiento y test sin validaciones

Como resultado de la ejecución del programa de este modo se obtendrá un único mejor individuo.

3.3.2 Individuo inicial

La fase con la que comienza su ejecución el algoritmo de búsqueda local desarrollado es con la creación de un individuo inicial.

Los individuos utilizados durante la ejecución del algoritmo implementado son representados mediante matrices.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \cdots & & & & \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}$$

Imagen 23: Formato de los individuos

Cada uno de los elementos que forman el individuo, a_{ij}, se empleará para modificar los atributos que componen las instancias de los datos utilizados durante las fases de entrenamiento y test.

La implementación desarrollada ha considerado cuatro tipos de individuos que serán los utilizados durante toda la ejecución, manteniendo siempre su tipo. Los cuatro tipos considerados son los siguientes:

- Individuo general.
- Individuo diagonal.
- Individuo simétrico.
- Individuo identidad.

Tal y como se ha descrito con anterioridad, se pueden emplear diferentes tipos de individuos, permitiendo al usuario ejecutar la aplicación con hasta cuatro tipos de individuos simultáneamente dependiendo de las necesidades o requerimientos del problema.

El individuo principal con el que se inicializan los individuos de cada tipo, independientemente del tipo, es una matriz identidad o lo que se denominará individuo "euclideo".

3.3.2.1 Individuo completo

Este tipo de individuo se caracteriza porque cada uno de los elementos, a_{ij}, que componen la matriz que representa al individuo puede tomar cualquier valor.

$$G = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \cdots & & & & \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}$$

Imagen 24: Individuo completo

3.3.2.2 Individuo diagonal

Este tipo de individuo se caracteriza porque cada uno de los elementos, a_{ij}, que componen la matriz que representa al individuo será 0; excepto si i = j, donde el elemento puede tomar cualquier valor.

$$D = \begin{pmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ 0 & a_{22} & 0 & \cdots & 0 \\ \cdots & & & & \\ 0 & 0 & 0 & \cdots & a_{nn} \end{pmatrix}$$

Imagen 25: Individuo diagonal

3.3.2.3 Individuo simétrico

Este tipo de individuo se caracteriza porque cada uno de los elementos, a_{ij}, que componen la matriz que representa al individuo puede tomar cualquier valor, siempre y cuando $a_{ij} = a_{ji}$.

$$S = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \cdots & & & & \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}$$

Imagen 26: Individuo simétrico

NOTA: Recordar que es necesario que $a_{ij} = a_{ji}$.

3.3.2.4 Individuo identidad

Este tipo de individuo se caracteriza porque cada uno de los elementos, a_{ij}, que componen la matriz que representa al individuo será 0; excepto si i = j, donde el elemento puede tomará el valor 1.

$$I = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \cdots & & & & \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

Imagen 27: Individuo identidad

3.3.3 Fitness del "individuo principal"

Una vez creado el individuo principal se procede a evaluar dicho individuo y conocer cómo de bueno es. Para ello se procede a asignarle al individuo un valor de fitness.

El individuo principal tendrá dos valores de fitness:

- **Fitness de entrenamiento**. Este valor de fitness servirá para verificar que el individuo principal va evolucionando a lo largo de la ejecución del algoritmo de búsqueda local. Durante la evolución este valor irá creciendo y ratificando que el individuo va evolucionando. Se utiliza el conjunto de entrenamiento para calcular este valor de fitness.

- **Fitness de test**. Este valor de fitness se calculará una vez el individuo principal ha dejado de evolucionar y por lo tanto ya se ha encontrado el mejor individuo. Para el cálculo de este valor de fitness se utilizará el conjunto destinado a test, es decir, un conjunto de datos que el algoritmo K-NN no haya visto ni utilizado para el entrenamiento.

El valor de fitness de entrenamiento de un individuo se obtiene de la siguiente manera:

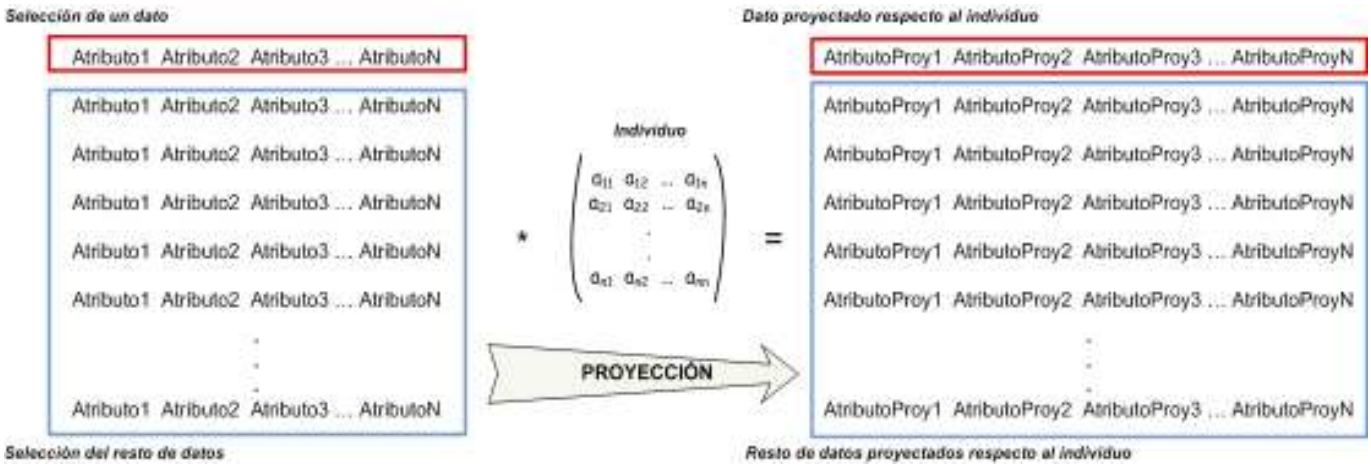


Imagen 28: Proyección de los datos

1. Se selecciona un dato del conjunto de entrenamiento y se proyecta respecto a la matriz del individuo.
2. Se proyectan el resto de datos de entrenamiento respecto al mismo individuo.
3. Se aplica el algoritmo K-NN para clasificar el dato seleccionado ya proyectado respecto al resto de datos también proyectados, y se contabiliza si K-NN acierta la clase de salida o no, dado que es un algoritmo supervisado.
4. Se vuelve al paso 1, seleccionando otro dato, hasta que se seleccionen todos los datos de entrenamiento.
5. El valor del fitness de entrenamiento que se le asignará al individuo será el siguiente:

$$\text{fitness} = \frac{\text{nº de datos acertados por K-NN}}{\text{nº de datos totales de entrenamiento}}$$

Imagen 29: Fitness de un individuo

3.3.4 Generar individuos descendientes

La generación de los individuos descendientes se realizará mediante la mutación de los elementos del individuo principal. Mediante estas mutaciones del individuo principal el algoritmo programado de búsqueda local tratará de ir mejorando el porcentaje de éxito en su clasificación.

En el programa creado se han implementado dos formas diferentes de realizar la generación de los individuos descendientes del individuo principal.

Se ha tenido en cuenta a la hora de implementar el programa, y se permite que el usuario pueda ejecutar el programa con las dos formas de generación de descendientes de forma independiente o de forma conjunta, ejecutándose los dos métodos paralelamente.

A continuación se describen los dos tipos de generación de descendientes implementados y se indican los parámetros requeridos por cada uno de ellos para configurar la ejecución del mismo (Para más información ver *Manual de Usuario*).

3.3.4.1 Evolución normal

Esta forma de evolución del individuo principal es similar a realizar una búsqueda en amplitud.

A partir del individuo principal se genera un número de hijos igual al número total de elementos que contiene la matriz que representa al individuo.

En cada uno de los hijos generados se mutará únicamente un elemento. La mutación de este elemento se producirá sumándole al elemento un valor x adquirido mediante una distribución normal N(0, σ). El valor de la desviación típica σ es establecido por el usuario en la pantalla inicial de configuración del programa.

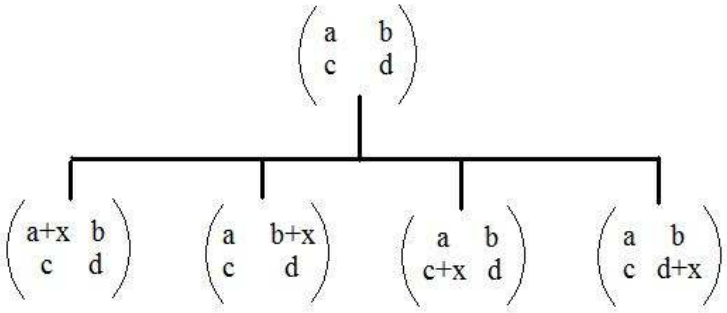


Imagen 30: Evolución normal del individuo principal

Importante destacar que el valor x que aparece en la imagen (para ser sumado a los elemento) no tiene porque ser el mismo en todas la mutaciones. Este valor x se genera aleatoriamente mediante N(0, σ).

3.3.4.2 Evolución "N Hijos"

Mediante este tipo de evolución se pueden mutar todos los elementos del individuo principal en el proceso de evolución. Como resultado de este proceso se genera un único hijo con los elementos mutados.

La mutación de los elementos dependerá de una probabilidad de mutación que el usuario debe indicar en la pantalla inicial de configuración del programa. El atributo aparece en la pantalla nombrado como *Probabilidad* en la sección *Parámetros de mutación.*

Para llevar a cabo la mutación de los elementos, al igual que el caso anterior, se utiliza una distribución Normal N(0,σ). El valor de σ será introducido por el usuario en la pantalla de configuración.

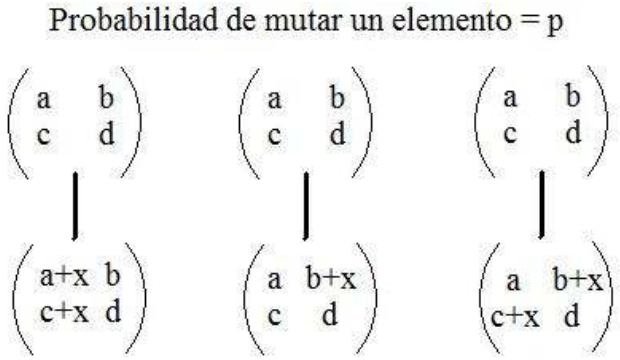


Imagen 31: Diferentes ejemplos de evolución "hijo único"

Importante destacar que el valor x que aparece en la imagen (para ser sumado a los elemento) no tiene porque ser el mismo en todas la mutaciones. Este valor x se genera aleatoriamente mediante N(0, σ).

Este tipo de evolución se caracteriza por generar únicamente un hijo. Para ajustar la evolución e impedir que ésta finalice rápidamente se da la posibilidad de repetir este proceso un número determinado de veces.

Cuando se genera el hijo se obtiene el fitness del mismo (siguiente apartado). Una vez obtenido este fitness se procede a comparar el fitness del individuo principal con el del hijo generado. Si el hijo generado no mejora el porcentaje de aciertos (fitness) respecto al del individuo principal se procede a generar un nuevo hijo.

El número máximos de hijos a generar puede ser configurado por el usuario en la pantalla de configuración en *Número de hijos* de la sección *Parámetros de mutación.*

3.3.5 Fitness de los individuos descendientes

El proceso para calcular el fitness (fitness de entrenamiento) de los individuos descendientes es similar al procedimiento utilizado para calcular el fitness de entrenamiento del individuo principal.

Los pasos son siguientes:

1. Se selecciona un dato del conjunto de entrenamiento y se proyecta respecto a la matriz del individuo.
2. Se proyectan el resto de datos de entrenamiento respecto al mismo individuo.
3. Se aplica el algoritmo K-NN para clasificar el dato seleccionado ya proyectado respecto al resto de datos también proyectados, y se contabiliza si K-NN acierta la clase de salida o no, dado que es un algoritmo supervisado.
4. Se vuelve al paso 1, seleccionando otro dato, hasta que se seleccionen todos los datos de entrenamiento.
5. El valor del fitness de entrenamiento que se le asignará al individuo será el siguiente:

$$\text{fitness} = \frac{\text{nº de datos acertados por K-NN}}{\text{nº de datos totales de entrenamiento}}$$

Imagen 32: Fitness del individuo descendiente

Este proceso se repetirá tantas veces como hijos descendientes hayan sido generados en el proceso de generación de individuos descendientes.

3.3.6 Fin de la evolución

Los individuos descendientes son generados y se les asigna el valor de fitness de correspondiente al porcentaje de aciertos con los datos del conjunto de entrenamiento.

Una vez llegados a este punto es necesario establecer un criterio de parada para detener la evolución del individuo.

Los siguientes apartados describen los mecanismos desarrollados en cada uno de los dos modos implementados para detener la evolución del individuo y obtener un resultado para el problema que presentan los datos introducidos.

3.3.6.1 Evolución normal

Este tipo de evolución, como se ha descrito anteriormente, genera un número de hijos igual al número total de elementos que componen el individuo.

La forma de detener el algoritmo es mediante la comparación del fitness del individuo principal con el de todos los hijos. Si existe algún hijo que tenga un fitness superior al de su "padre", se seleccionará el que posea mayor valor de fitness para sustituirlo y continuar con la ejecución del algoritmo.

Por el contrario, si no existe ningún descendiente que supere el valor de fitness de su progenitor se detiene el funcionamiento del algoritmo. En este caso el "mejor individuo" resultante es el progenitor de estos últimos descendientes.

3.3.6.2 Evolución "N Hijos"

La evolución con "hijo único" crea un solo descendiente a partir de mutaciones del individuo principal. Este hijo puede tener mutados cualquiera de sus elementos, de acuerdo a una probabilidad de mutación indicada por el usuario.

La detección del proceso de evolución del individuo es similar al caso anteriormente descrito. Se compara el fitness del descendiente con el del progenitor. Si el valor de fitness del descendiente es mayor al de su "padre", se sustituirá el progenitor por el descendiente y se continuará con el normal funcionamiento del algoritmo.

Sin embargo, si el fitness del descendiente generado no supera el valor de fitness de su "padre" se genera un nuevo individuo descendiente a partir del mismo individuo progenitor. Este proceso se realizará un número de veces indicado por el usuario en la ventana de configuración inicial. Si alguno de los descendientes generados supera el fitness de su padre, se continúa con el funcionamiento normal; y si por el contrario ninguno consigue superarlo, se aborta la evolución del algoritmo y se devuelve el individuo principal como mejor resultado obtenido.

3.3.7 Fin del algoritmo

Una vez el individuo ha finalizado su evolución es el momento de comprobar si el algoritmo implementado ha alcanzado su fin. Dependiendo del modo de funcionamiento utilizado existen dos maneras de comprobar la finalización del algoritmo:

- **Usando validaciones**. Cuando se alcanza la finalización de la evolución del individuo principal se comprueba si se han utilizado para test todas las folders. Si se han utilizado todas las folders para test, entonces se finaliza el algoritmo; sino, se generan los nuevos conjuntos de entrenamiento y test variando los folders utilizados para ello, y se vuelve a ejecutar el algoritmo normalmente.

- **Usando ficheros con datos de entrenamiento y test**. En este caso, el algoritmo solo se ejecuta una única vez con los conjuntos de entrenamiento y test indicados mediante ficheros. El algoritmo finaliza al mismo tiempo que finaliza la evolución del individuo principal.

Al finalizar el funcionamiento del algoritmo, se mostrarán en el entorno de ejecución un breve resumen de los resultados obtenidos. Pero, para obtener mayor detalle se pueden analizar los ficheros que genera el programa desarrollado.

Estos ficheros se crearán en el directorio que el usuario indique a través de la interfaz del programa. El árbol de directorios generados es el siguiente:

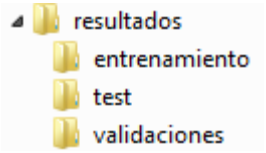


Imagen 33: Directorio de los datos de salida

Como se puede observar, se generan los siguientes subdirectorios:

- **entrenamiento**. Directorio que contiene ficheros para cada tipo de búsqueda, conteniendo información sobre la evolución del individuo a lo largo de la ejecución del programa. Si se han usado validaciones se crearán tantos ficheros como validaciones por cada tipo de búsqueda seleccionado.

  Descripción de las líneas que contienen los ficheros:

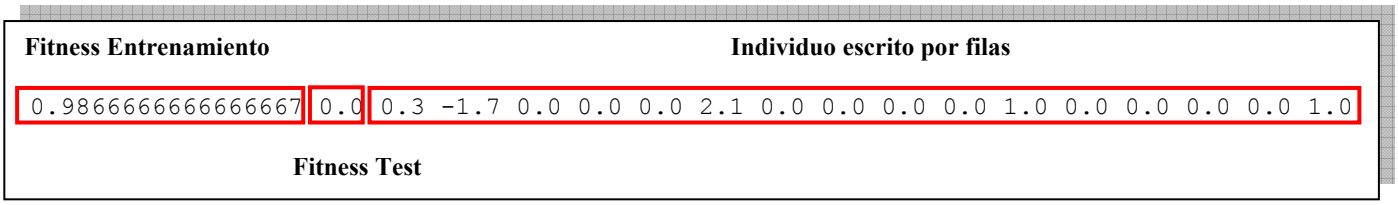


Imagen 34: Descripción de los ficheros de generados en entrenamiento

- **test**. Directorio que contiene ficheros para cada tipo de búsqueda, conteniendo información sobre el resultado obtenido por el individuo en la fase de test. Si se han usado validaciones se crearán tantos ficheros como validaciones por cada tipo de búsqueda seleccionado.

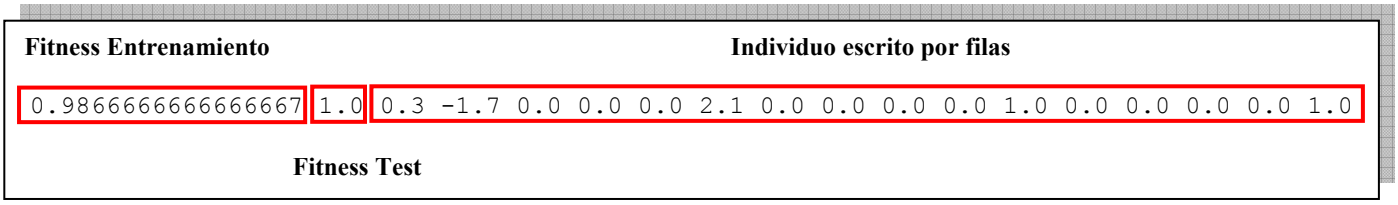


Imagen 35: Descripción de los ficheros de generados en entrenamiento

- **validaciones**. Directorio que contiene ficheros con las diferentes validaciones generadas. Cada fichero contendrá los datos que aleatoriamente se hayan seleccionado para cada validación.

  Este directorio únicamente se generará si el usuario ha indicado que se utilicen validaciones. Se generarán tantos ficheros como validaciones se haya indicado.

Capítulo 4

Experimentación

4.1 Introducción

El valor aportado por el proyecto se ha obtenido en la fase de experimentación, en la que se han ido obteniendo resultados con el programa implementado.

En esta fase del desarrollo del proyecto se han utilizado diferentes dominios de datos para probar el funcionamiento del programa y poder obtener finalmente unos resultados. Los resultados obtenidos nos ayudarán a evaluar el algoritmo de búsqueda local.

En este capítulo se presentan los resultados obtenidos para una serie de dominios de datos seleccionados. Los resultados obtenidos serán comparados entre sí, extrayendo resultados que se utilizarán para extraer conclusiones.

Los dominios utilizados en la fase de experimentación han sido los siguientes:

- Ripley.
- Diabetes.
- Iris.
- Yeast.
- EEG.
- Dominios sintéticos.

A continuación se describirán cada uno de estos dominios, indicando los atributos que componen los datos de los mismos.

4.2 Dominio Ripley

El dominio de datos Ripley contiene instancias que han sido generadas artificialmente. Cada una de las instancias cuenta con dos valores reales como coordenadas y una clase que puede 0 o 1. La generación de la clase corresponde a una distribución bimodal que es una composición equilibrada de dos distribuciones. El único cambio referente a las dos distribuciones es el centro de ambas.

Información acerca de los datos utilizados:

Instancias totales	Atributos	Clases
1250	2	2

Tabla 1: Características del dominio Ripley

Nota: Los atributos no incluyen el atributo que describe la clase de la instancia.

Los atributos que componen las instancias del dominio Ripley son:

1. Coordenada x (número real).
2. Coordenada y (número real).
3. Variable de clase :
   - 0
   - 1

Debido a que el dominio Ripley está formado por dos coordenadas, se ha podido representar gráficamente el conjunto de datos utilizados. A continuación se presenta el mencionado conjunto:

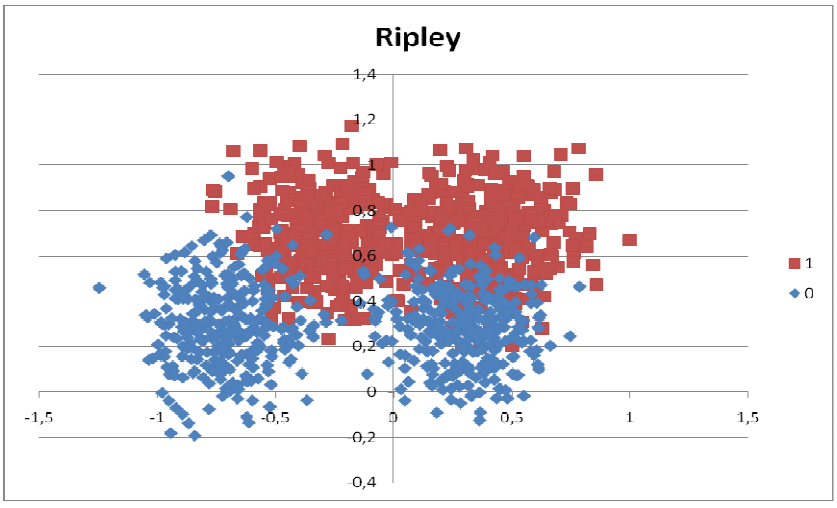


Imagen 36: Distribución de los datos del dominio Ripley

La distribución presentada anteriormente contiene datos referentes al dominio obtenido de dos ficheros de datos, uno con datos para entrenamiento y otro para test, que han sido mezclados en un único fichero. Por este motivo se han representado todos los datos de forma conjunta.

Las pruebas realizadas para comprobar el funcionamiento del algoritmo desarrollado en el dominio Ripley han sido realizadas con la siguiente configuración:

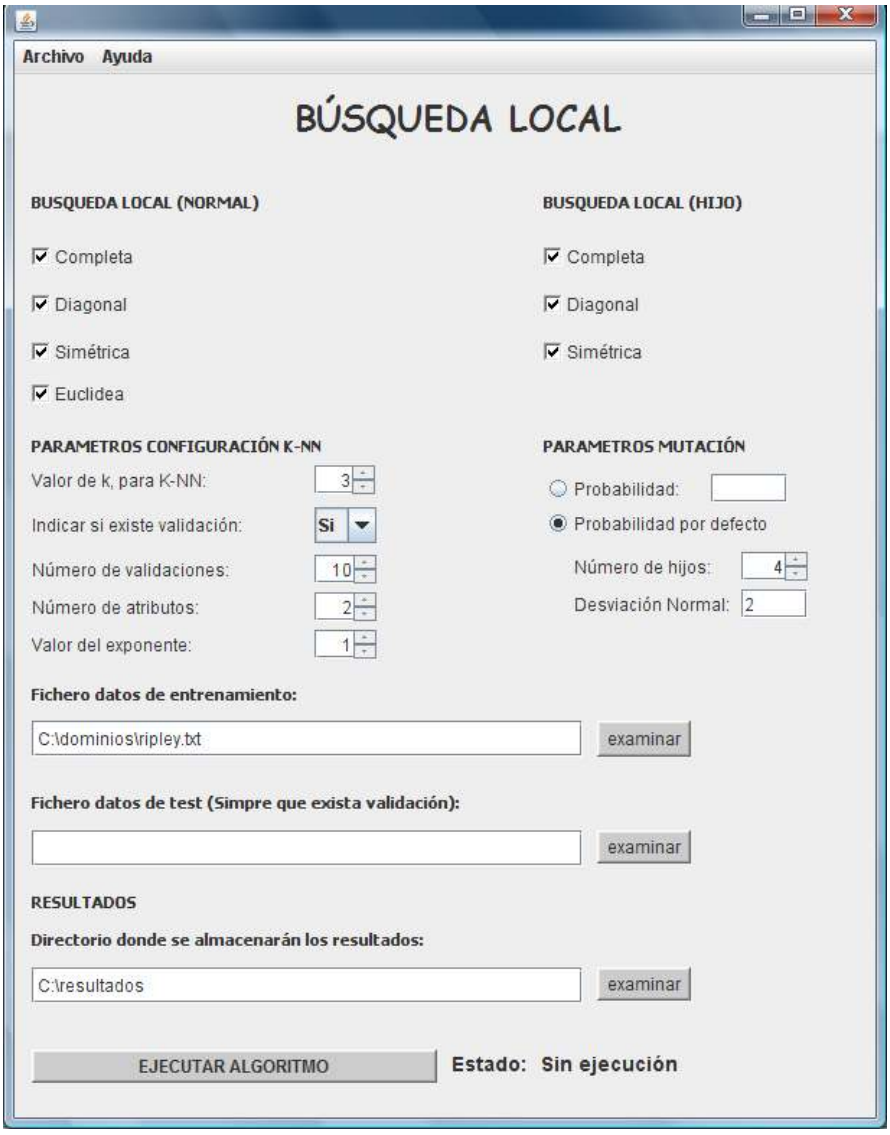


Imagen 37: Configuración para las pruebas de Ripley

Según la configuración anterior, se han realizado 3 validaciones cruzada de 10 hojas. En cada validación cruzada se desordenan los datos antes de particionarlos en las 10 hojas. El objetivo de repetir 3 veces la validación cruzada es comprobar la variabilidad de los resultados, de manera que las conclusiones sean más fiables.

El número de hijos establecido es 4. Este número de hijos será el número de atributos del dominio elevado al cuadrado.

Los resultados de cada una de estas pruebas se muestran a continuación.

Prueba 1

Resultados de la primera prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**
1	0,9120	0,9120	0,9360	0,9360	0,9360	0,8960	0,9360
2	0,9200	0,9200	0,9200	0,9200	0,9200	0,9200	0,9200
3	0,9120	0,9040	0,9120	0,8640	0,8880	0,8960	0,9040
4	0,9200	0,9200	0,9200	0,9200	0,8960	0,9200	0,9200
5	0,9360	0,9600	0,9280	0,9280	0,9520	0,9520	0,9280
6	0,8960	0,8560	0,8640	0,8880	0,8880	0,8880	0,8640
7	0,9360	0,9360	0,9360	0,9360	0,9360	0,9360	0,9360
8	0,8560	0,8560	0,8640	0,8560	0,8640	0,8560	0,8640
9	0,9120	0,9120	0,9440	0,9360	0,9440	0,9280	0,9440
10	0,9600	0,9680	0,9680	0,9680	0,9680	0,9680	0,9680
Media	**0,916**	**0,9144**	**0,9192**	**0,9152**	**0,9192**	**0,916**	**0,9184**

Tabla 2: Resultados de la prueba 1 de Ripley

En la tabla se muestra a la izquierda cual de las 10 validaciones obtenidas ha sido utilizada como conjunto de test.

Además, se observan los resultados obtenidos para cada tipo de matriz: completa, diagonal, simétrica e identidad. Para cada tipo de matriz se han utilizado dos modos de ejecución:

- **Búsqueda normal**. La mutación de una matriz (o individuo) genera un número de hijos igual a la dimensión de la matriz al cuadrado. Por ejemplo, si el individuo es de dimensión tres, la mutación generará nueve hijos. Cada hijo solo cambiará un elemento de la matriz respecto a su progenitor.

- **Búsqueda con N hijos**. La mutación de un individuo genera un nuevo hijo, en el que todos los elementos pueden variar su valor de acuerdo a la probabilidad de mutación indicada por el usuario. Si no mejora el fitness del padre, generará otro y así hasta alcanzar el máximo número de hijos establecido por el usuario en la opción correspondiente de la pantalla de configuración.

Prueba 2

Resultados de la segunda prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**
1	0,9200	0,9200	0,9200	0,9200	0,9200	0,9200	0,9200
2	0,9360	0,9360	0,9360	0,9360	0,9360	0,9360	0,9360
3	0,9520	0,9520	0,9360	0,9360	0,9360	0,9200	0,9360
4	0,9600	0,9440	0,9360	0,9440	0,9600	0,9520	0,9440
5	0,9280	0,9280	0,9280	0,9440	0,9360	0,9280	0,9280
6	0,9280	0,9360	0,9280	0,9280	0,9360	0,9280	0,9280
7	0,9360	0,9520	0,9520	0,9520	0,9520	0,9520	0,9520
8	0,9280	0,9200	0,9280	0,9280	0,9200	0,9280	0,9280
9	0,9280	0,9280	0,9200	0,9200	0,9200	0,9280	0,9200
10	0,9520	0,9600	0,9600	0,9600	0,9520	0,9600	0,9600
Media	**0,9368**	**0,9376**	**0,9344**	**0,9368**	**0,9368**	**0,9352**	**0,9352**

Tabla 3: Resultados de la prueba 2 de Ripley

Prueba 3

Resultados de la tercera prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**
1	0,9520	0,9600	0,9600	0,9520	0,9520	0,9600	0,9600
2	0,8960	0,8480	0,8880	0,8960	0,8880	0,9040	0,8880
3	0,9760	0,9760	0,9360	0,9280	0,9440	0,9280	0,9280
4	0,9360	0,9360	0,9360	0,9440	0,9280	0,9200	0,9360
5	0,9680	0,9440	0,9600	0,9600	0,9520	0,9440	0,9600
6	0,9440	0,9440	0,9520	0,9440	0,9280	0,9600	0,9600
7	0,9360	0,9360	0,9200	0,9360	0,9360	0,9360	0,9360
8	0,9280	0,9200	0,9200	0,9200	0,9360	0,9360	0,9200
9	0,9200	0,9200	0,9120	0,9440	0,9280	0,9280	0,9200
10	0,9520	0,9520	0,9520	0,9520	0,9520	0,9600	0,9520
Media	**0,9408**	**0,9336**	**0,9336**	**0,9376**	**0,9344**	**0,9376**	**0,936**

Tabla 4: Resultados de la prueba 3 de Ripley

El resumen de las pruebas muestra los siguientes resultados (Tabla 5):

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
Media total	0,9312	0,92853333	0,92906667	0,92986667	0,93013333	0,9296	0,929866667

Tabla 5: Resumen de resultados de las pruebas de Ripley

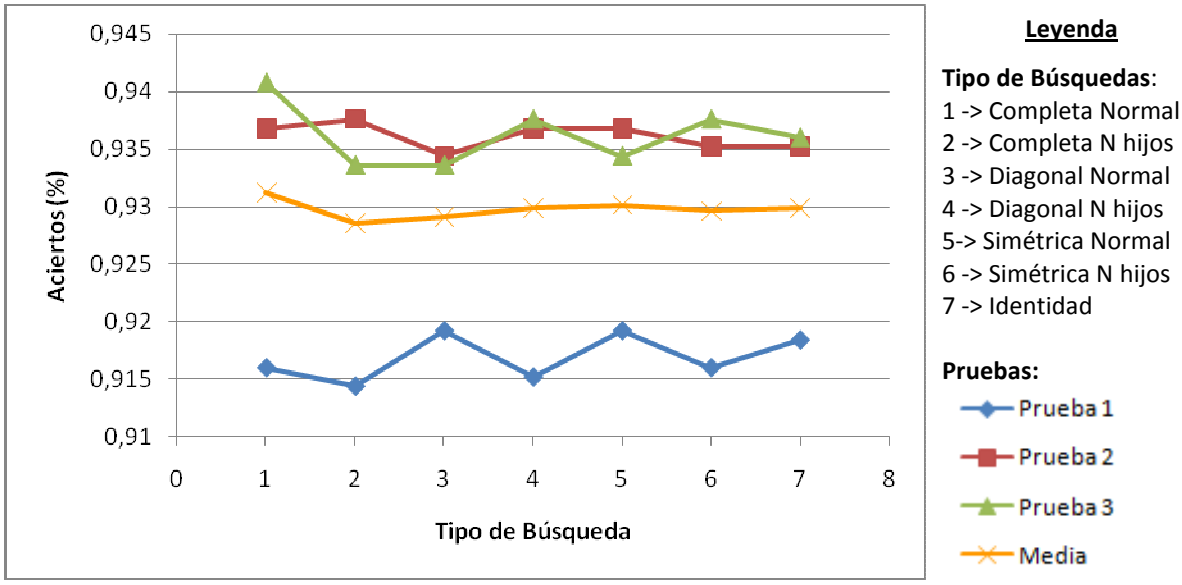


Imagen 38: Representación gráficas de las pruebas para Ripley

Observando la representación gráfica de todas la pruebas (Imagen 38), se puede observar cómo en media no hay apenas diferencias entre los diferentes tipos de matrices. El porcentaje de aciertos para todos los casos se encuentra alrededor del 93%.

Para este caso, se puede afirmar que ninguna de las diferentes búsquedas aporta ningún valor significativo al obtenido sobre la matriz identidad.

4.3 Dominio Diabetes

El dominio Diabetes contiene datos que fueron seleccionados de una base de datos mayor. Concretamente, los datos que se han utilizado corresponden a pacientes que son mujeres con al menos 21 años. Además, todas estas mujeres pertenecen a los Indios PIMA.

Información acerca de los datos utilizados:

Instancias totales	Atributos	Clases
768	**8**	**2**

Tabla 6: Características del dominio Diabetes

Nota: Los atributos no incluyen el atributo que describe la clase de la instancia.

Los atributos que componen las instancias del dominio Diabetes son:

1. Número de veces embarazada.
2. Concentración de glucosa en plasma 2 horas en una prueba de tolerancia oral a la glucosa.
3. Presión arterial diastólica (mmHg).
4. Espesor del pliego tricipital (mm).
5. 2 horas de insulina en suero (muU/ml).
6. Índice de masa corporal (kg/m^2).
7. Diabetes función pedigrí.
8. Edad (años).
9. Variable de clase :
    - 0
    - 1

Las pruebas realizadas para comprobar el funcionamiento del algoritmo desarrollado en el dominio Diabetes han sido realizadas con la siguiente configuración:

Imagen 39: Configuración para las pruebas de Diabetes

Al igual que el caso anterior, para obtener unos resultados más fiables, se han realizado tres pruebas con la misma configuración. En este caso, se utilizarán 8 atributos, que son los que componen los datos del dominio, y al igual que antes se utilizará validación cruzada de 10 hojas.

Se realizan todos los tipos de búsquedas para poder comparar posteriormente los resultados obtenidos.

Prueba 1

Resultados de la primera prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**
1	0,7631579	0,8552632	0,7894737	0,8684211	0,8421053	0,75	0,7631579
2	0,8026316	0,7236842	0,7105263	0,7763158	0,75	0,6842105	0,7631579
3	0,8421053	0,8289474	0,7894737	0,8026316	0,7894737	0,7894737	0,7236842
4	0,7763158	0,8157895	0,8421053	0,8684211	0,8684211	0,8815789	0,8684211
5	0,8421053	0,8815789	0,7631579	0,7894737	0,8421053	0,8684211	0,8552632
6	0,8289474	0,7763158	0,7763158	0,8289474	0,7894737	0,8026316	0,7894737
7	0,7368421	0,6973684	0,8157895	0,7105263	0,75	0,6973684	0,75
8	0,8552632	0,8552632	0,8289474	0,75	0,8157895	0,7894737	0,7894737
9	0,8026316	0,8421053	0,8289474	0,8026316	0,7763158	0,8026316	0,8026316
10	0,7236842	0,8026316	0,7763158	0,6973684	0,7631579	0,6973684	0,7631579
Media	**0,79736842**	**0,80789474**	**0,79210526**	**0,78947368**	**0,79868421**	**0,77631579**	**0,78684211**

Tabla 7: Resultados de la prueba 1 de Diabetes

Prueba 2

Resultados de la segunda prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**
1	0,8157895	0,8552632	0,8421053	0,8157895	0,8421053	0,7894737	0,8026316
2	0,7631579	0,8157895	0,7631579	0,7763158	0,8157895	0,7236842	0,75
3	0,8421053	0,8421053	0,8421053	0,8289474	0,8157895	0,8157895	0,7894737
4	0,7236842	0,7368421	0,7105263	0,6973684	0,75	0,6710526	0,7236842
5	0,8026316	0,8289474	0,8552632	0,8289474	0,8421053	0,8421053	0,8026316
6	0,8026316	0,75	0,8026316	0,8289474	0,7894737	0,7631579	0,7631579
7	0,6973684	0,75	0,7631579	0,75	0,6710526	0,8289474	0,6710526
8	0,7763158	0,7368421	0,75	0,8157895	0,7631579	0,7631579	0,7368421
9	0,8026316	0,7236842	0,75	0,8157895	0,8026316	0,7631579	0,7368421
10	0,7894737	0,7368421	0,8289474	0,8421053	0,7894737	0,75	0,8289474
Media	**0,78157895**	**0,77763158**	**0,79078947**	**0,8**	**0,78815789**	**0,77105263**	**0,76052632**

Tabla 8: Resultados de la prueba 2 de Diabetes

Prueba 3

Resultados de la segunda prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**
1	0,75	0,7631579	0,7631579	0,7894737	0,75	0,8026316	0,7236842
2	0,8289474	0,8421053	0,8421053	0,7894737	0,8157895	0,8157895	0,8289474
3	0,7894737	0,8289474	0,7763158	0,8157895	0,7631579	0,7105263	0,7763158
4	0,8026316	0,8552632	0,8552632	0,8289474	0,7763158	0,8815789	0,8026316
5	0,8552632	0,8421053	0,8289474	0,8157895	0,8157895	0,8026316	0,7894737
6	0,7368421	0,6842105	0,6842105	0,7368421	0,7105263	0,6578947	0,6710526
7	0,7763158	0,75	0,7631579	0,8684211	0,7631579	0,7763158	0,75
8	0,8947368	0,8289474	0,7894737	0,8421053	0,8157895	0,8421053	0,8421053
9	0,75	0,7105263	0,7236842	0,75	0,75	0,7631579	0,7368421
10	0,8421053	0,8157895	0,8289474	0,8157895	0,8026316	0,8552632	0,7631579
Media	**0,80263158**	**0,79210526**	**0,78552632**	**0,80526316**	**0,77631579**	**0,79078947**	**0,76842105**

Tabla 9: Resultados de la prueba 3 de Diabetes

El resumen de las pruebas muestra los siguientes resultados (Tabla 10):

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
Media	0,79385965	0,79254386	0,78947368	0,79824561	0,7877193	0,77938596	0,77192982

Tabla 10: Resumen de resultados de las pruebas de Diabetes

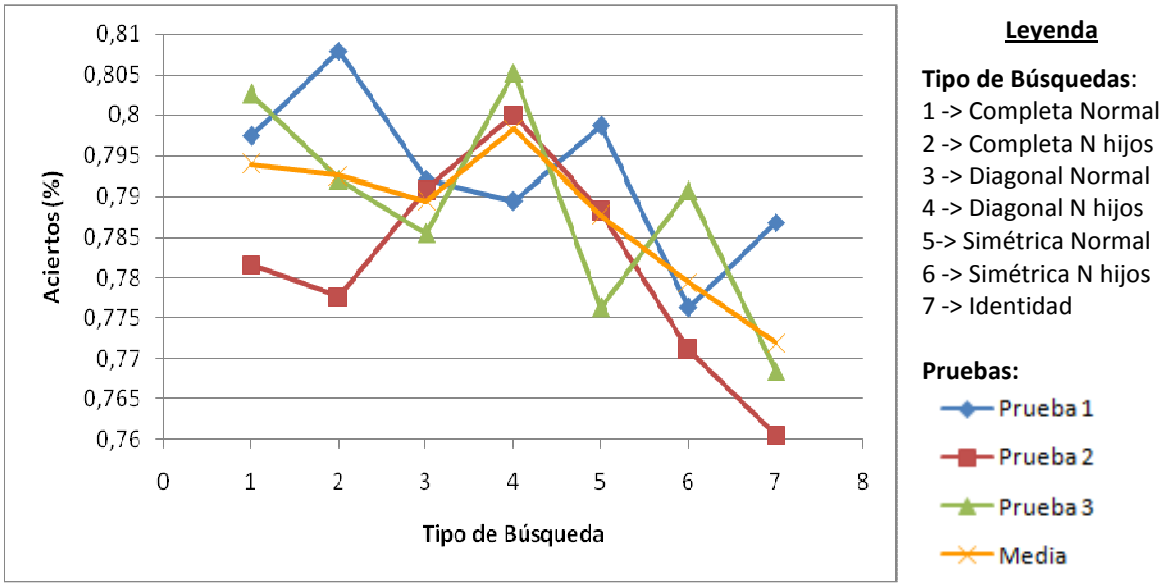


Imagen 40: Representación gráfica de las pruebas de Diabetes

Observando los resultados obtenidos con la media de todas las pruebas se observa cómo todos los tipos de matrices mejoran el rendimiento de la matriz identidad.

Todos los tipos de búsquedas, en al menos uno de sus modos de ejecución (normal o con N hijos), han conseguido superar en más de un 1% el porcentaje de aciertos de la matriz identidad.

La búsqueda con la matriz completa es muy regular, y obtiene buenos resultados comparados con la matriz identidad. En media, para ambos modos de ejecución, supera en más de un 2% el porcentaje de aciertos del caso base.

Finalmente y como resultado obtenido con este dominio, indicar que la matriz que mejores resultados ha obtenido ha sido la matriz diagonal en su modo de ejecución N hijos. El porcentaje de aciertos obtenido en la fase de test mejora en casi un 3% el porcentaje de la matriz identidad

4.4 Dominio Iris

El dominio Iris contiene un conjunto de datos con 3 clases de 50 casos cada uno, donde cada una de ellas se refiere a un tipo de planta del iris.

Información acerca de los datos utilizados:

Instancias totales	Atributos	Clases
150	**4**	**3**

Tabla 11: Características del dominio Iris

Nota: Los atributos no incluyen el atributo que describe la clase de la instancia.

Los atributos que componen las instancias del dominio Iris son:

1. Longitud del sépalo (cm).
2. Ancho del sépalo (cm).
3. Longitud del pétalo (cm).
4. Ancho del pétalo (cm).
5. Variable de clase:
    - Iris Setosa.
    - Iris Versicolour.
    - Iris Virginica.

Las pruebas realizadas para comprobar el funcionamiento del algoritmo desarrollado en el dominio Iris han sido realizadas con la siguiente configuración:

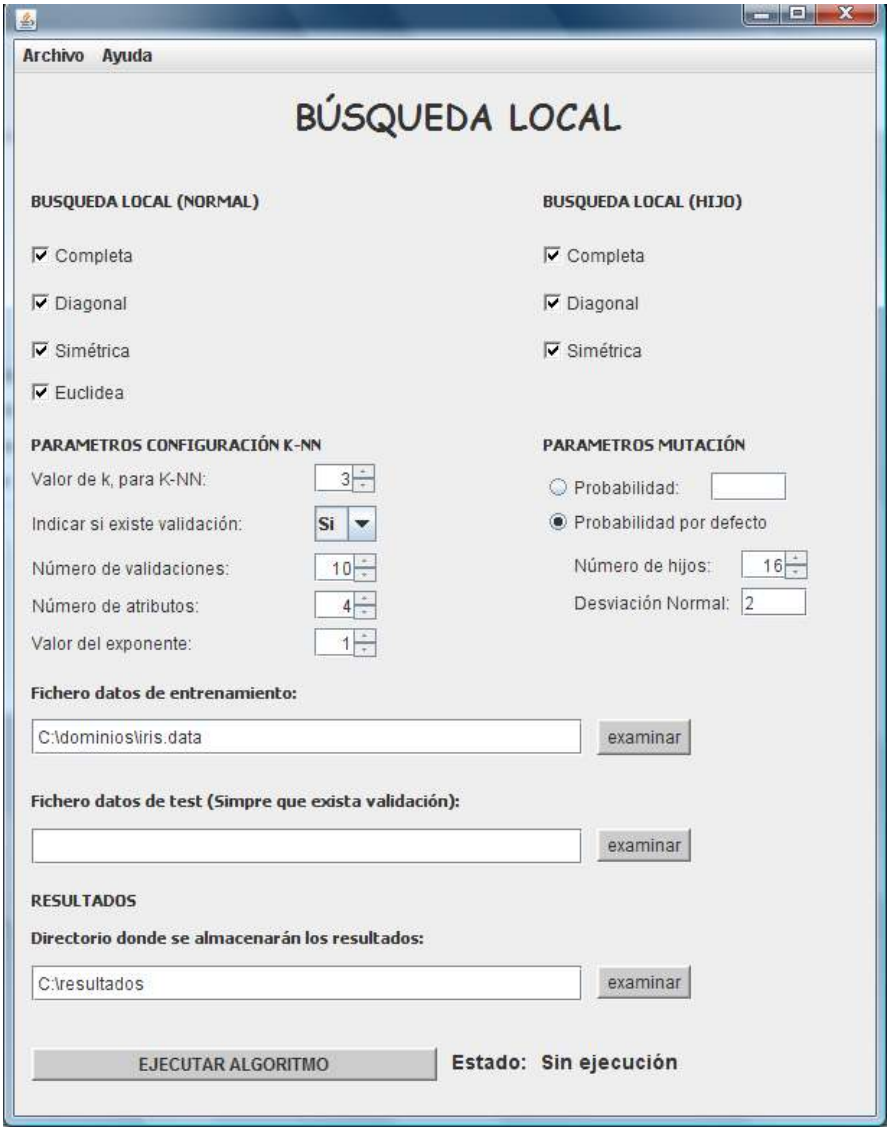


Imagen 41: Configuración para las pruebas de Iris

En este caso, se ajusta el número de atributos a 4, y se mantienen el resto de parámetros a excepción del número de hijos, que se considera 16 (número de atributos al cuadrado).

Prueba 1

Resultados de la primera prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**
1	0,9333	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
2	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
3	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
4	0,9333	0,9333	0,9333	0,9333	0,9333	0,9333	0,9333
5	0,9333	0,9333	0,9333	0,9333	0,9333	1,0000	0,9333
6	0,9333	0,9333	0,9333	0,9333	0,9333	0,9333	0,9333
7	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
8	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
9	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
10	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
Media	**0,97333333**	**0,98**	**0,98**	**0,98**	**0,98**	**0,98666667**	**0,98**

Tabla 12: Resultados de la prueba 1 de Iris

Prueba 2

Resultados de la segunda prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**
1	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
2	0,9333	0,8667	0,9333	1,0000	0,9333	0,9333	0,9333
3	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
4	0,9333	0,9333	0,9333	1,0000	0,9333	1,0000	1,0000
5	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
6	0,9333	0,9333	0,9333	0,9333	0,9333	1,0000	0,9333
7	0,9333	1,0000	0,9333	0,9333	0,9333	0,9333	0,9333
8	0,8667	1,0000	0,9333	1,0000	0,8667	0,9333	0,8667
9	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
10	1,0000	1,0000	1,0000	1,0000	1,0000	0,9333	1,0000
Media	**0,96**	**0,97333333**	**0,96666667**	**0,98666667**	**0,96**	**0,97333333**	**0,96666667**

Tabla 13: Resultados de la prueba 2 de Iris

Prueba 3

Resultados de la tercera prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**
1	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
2	1,0000	1,0000	0,9333	0,9333	1,0000	1,0000	0,9333
3	0,9333	0,9333	0,9333	1,0000	0,9333	0,9333	0,9333
4	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
5	0,9333	1,0000	0,9333	0,9333	0,9333	1,0000	0,9333
6	1,0000	1,0000	0,9333	0,9333	0,9333	0,9333	1,0000
7	0,9333	0,9333	1,0000	1,0000	0,9333	1,0000	0,9333
8	0,9333	0,9333	0,9333	1,0000	0,9333	0,9333	0,9333
9	0,8667	0,9333	0,8667	0,9333	0,8667	0,8667	0,8667
10	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
Media	**0,96**	**0,97333333**	**0,95333333**	**0,97333333**	**0,95333333**	**0,96666667**	**0,95333333**

Tabla 14: Resultados de la prueba 3 de Iris

El resumen de las pruebas muestra los siguientes resultados (Tabla 15):

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
Media	0,96444444	0,97555556	0,96666667	0,98	0,96444444	0,97555556	0,96666667

Tabla 15: Resumen de resultados de las pruebas de Iris

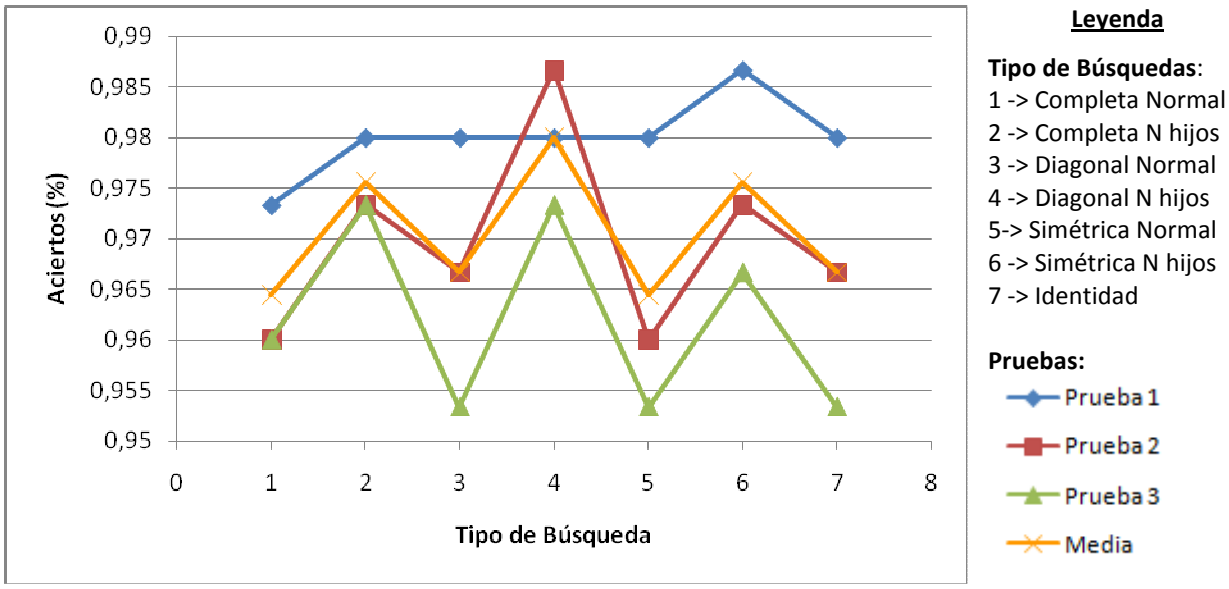


Imagen 42: Representación gráfica de las pruebas de Iris

Los resultados obtenidos en el conjunto de las tres pruebas demuestran que el tipo de matriz que mejor ha funcionado es la matriz diagonal con N hijos. Este tipo de matriz ha mejorado en casi un 1,5% los resultados de la matriz identidad y ha alcanzado en media un porcentaje de aciertos del 98%. Es interesante comprobar en la imagen 42 que los resultados de los distintos métodos en las distintas pruebas son bastante consistentes y que la matriz diagonal con N hijos tiene una tendencia clara a mejorar al resto de métodos y a la matriz identidad.

Otro dato que cabe destacar es el mejor funcionamiento del modo de obtención de descendientes N hijos. En este modo de ejecución las búsquedas han superado a su modo de ejecución normal. Todas lo superan en más del 1%.

Todos los valores de fitness de test obtenidos en las diferentes validaciones del conjunto de datos son “1”, “0,9333” o “0,8667”. Este hecho es producido por el número de instancias totales del dominio, 150. Al realizar 10 validaciones con esas instancias cada validación contará con 15 instancias. Los resultados obtenidos como fitness son producidos porque el algoritmo es capaz de clasificar correctamente las 15 instancias, 14 de las 15 instancias y 13 de las 15 instancias, respectivamente.

4.5 Dominio Yeast

El dominio Yeast contiene un conjunto de datos con información acerca del lugar que debe ocupar una proteína.

Información acerca de los datos utilizados:

Instancias totales	Atributos	Clases
1484	**8**	**10**

Tabla 16: Características del dominio Yeast

Nota: Los atributos no incluyen el atributo que describe la clase de la instancia.

Los atributos que componen las instancias del dominio Yeast son:

1. mcg: Método de McGeoch para el reconocimiento de secuencias de señales.
2. gvh: Método de von Heijne para el reconocimiento de secuencias de señales.
3. alm: Puntuación de la membrana ALOM.
4. mit: Puntuación del análisis discriminante del aminoácido contenido en la región N-terminal (20 residuos de longitud) de la mitocondria y proteínas no mitocondriales.
5. erl: Presencia de la subcadena "HDEL" (0 o 1).
6. pox: Señales peroxisomales de orientación en el C-terminal.
7. vac: Puntuación del análisis discriminante de amininoácidos en los vacuolares y en las proteínas extracelulares.
8. nuc: Puntuación discriminante del análisis de la localización nuclear de proteínas nucleares y no nucleares.
9. Variable de clase:
    - CYT (citosólido o citoesqueleto)
    - NUC (nuclear)
    - MIT (mitocondrial)
    - ME3 (proteína de membrana, no señal de N-terminal)
    - ME2 (proteína de membrana, señal no fragmentada)
    - ME1 (proteína de membrana, señal fracturada)

- EXC (extracelular)
- VAC (vacuolar)
- POX (peroxisomales)
- ERL (Lumen del retículo endoplasmático)

Las pruebas realizadas para comprobar el funcionamiento del algoritmo desarrollado en el dominio Yeast han sido realizadas con la siguiente configuración:

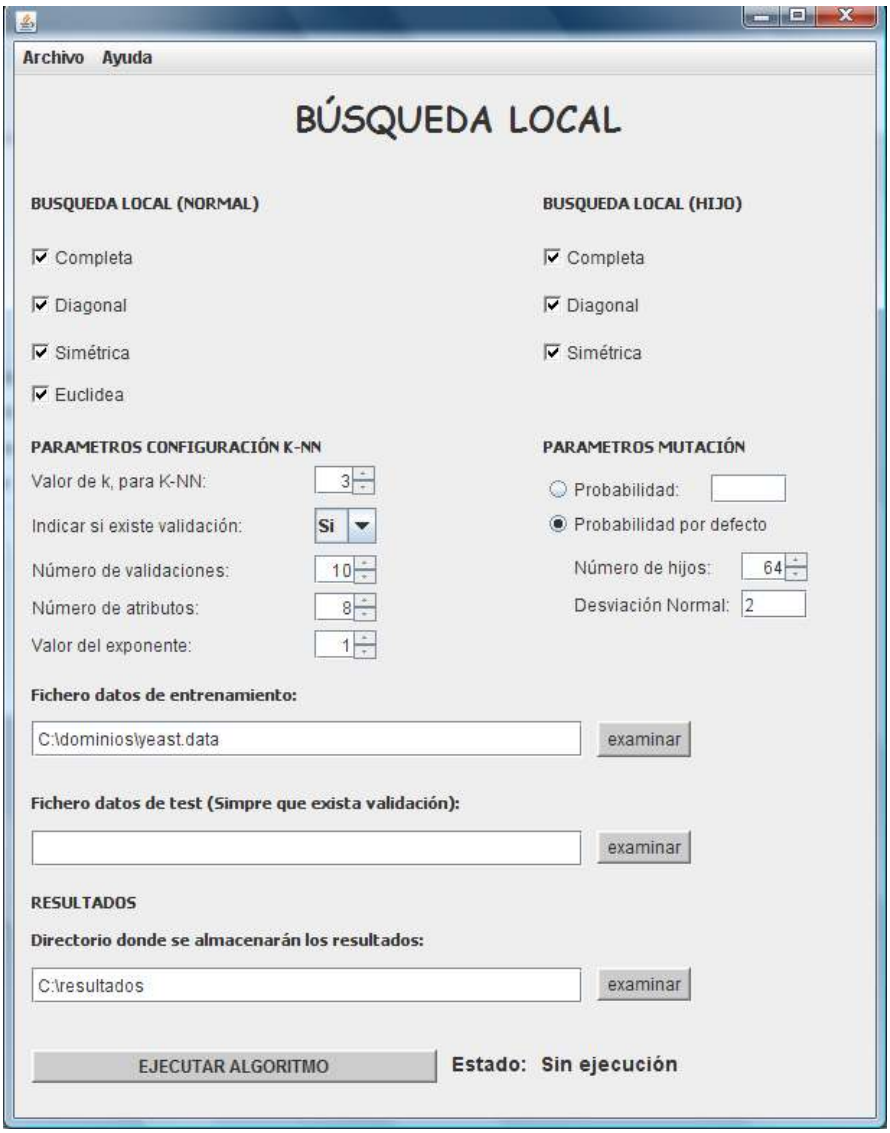


Imagen 43: Configuración para las pruebas de Yeast

Esta configuración, al igual que la utilizada en los dominios anteriores considera todos los tipos de búsquedas. Se utilizará validación cruzada de 10 hojas para obtener resultados más fiables, y el número de hijos máximo será de 64.

Prueba 1

Resultados de la primera prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**
1	0,7027027	0,7162162	0,6689189	0,7432432	0,7027027	0,7432432	0,7027027
2	0,6418919	0,7162162	0,6554054	0,6418919	0,6418919	0,6351351	0,6621622
3	0,6689189	0,7027027	0,6689189	0,6418919	0,6351351	0,6689189	0,6351351
4	0,7162162	0,6891892	0,6959459	0,6824324	0,6891892	0,6621622	0,6824324
5	0,6891892	0,7297297	0,7364865	0,6891892	0,7027027	0,6959459	0,7432432
6	0,6689189	0,6756757	0,6824324	0,7094595	0,6486486	0,6824324	0,6824324
7	0,6148649	0,6824324	0,6959459	0,722973	0,7027027	0,722973	0,7027027
8	0,7364865	0,6824324	0,722973	0,7094595	0,6756757	0,7297297	0,7094595
9	0,6689189	0,6959459	0,6148649	0,6554054	0,7027027	0,7027027	0,6824324
10	0,6621622	0,6621622	0,6689189	0,5810811	0,6756757	0,6554054	0,6486486
Media	**0,67702703**	**0,69527027**	**0,68108108**	**0,6777027**	**0,6777027**	**0,68986486**	**0,68513514**

Tabla 17: Resultados de la prueba 1 de Yeast

Prueba 2

Resultados de la segunda prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**
1	0,6283784	0,6148649	0,6554054	0,7027027	0,6756757	0,6554054	0,6756757
2	0,6283784	0,6148649	0,5878378	0,6351351	0,6351351	0,6216216	0,6554054
3	0,6554054	0,6486486	0,6486486	0,6283784	0,6418919	0,6689189	0,6351351
4	0,6418919	0,6216216	0,6756757	0,7297297	0,6891892	0,6486486	0,6486486
5	0,6689189	0,6486486	0,6891892	0,6283784	0,6756757	0,6283784	0,6283784
6	0,6756757	0,6486486	0,6621622	0,6891892	0,6621622	0,6621622	0,6689189
7	0,722973	0,6824324	0,6891892	0,7162162	0,7094595	0,6756757	0,7027027
8	0,6283784	0,6148649	0,6216216	0,6621622	0,6689189	0,6418919	0,5878378
9	0,6283784	0,6418919	0,6418919	0,6824324	0,6081081	0,6216216	0,6216216
10	0,6013514	0,6891892	0,6013514	0,5878378	0,6351351	0,6283784	0,6216216
Media	**0,64797297**	**0,64256757**	**0,6472973**	**0,66621622**	**0,66013514**	**0,64527027**	**0,64459459**

Tabla 18: Resultados de la prueba 2 de Yeast

Prueba 3

Resultados de la tercera prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**
1	0,6283784	0,6486486	0,6486486	0,6216216	0,6824324	0,6554054	0,6486486
2	0,6621622	0,6689189	0,6824324	0,6689189	0,6351351	0,6689189	0,6689189
3	0,6689189	0,722973	0,6756757	0,6689189	0,6554054	0,6756757	0,6689189
4	0,5945946	0,6418919	0,6216216	0,6689189	0,5878378	0,5878378	0,6216216
5	0,7094595	0,6959459	0,7027027	0,7297297	0,722973	0,6756757	0,7027027
6	0,6689189	0,6891892	0,6756757	0,6824324	0,6756757	0,6351351	0,6824324
7	0,6621622	0,6216216	0,6689189	0,6689189	0,6013514	0,6621622	0,6283784
8	0,6081081	0,6486486	0,6216216	0,6351351	0,6283784	0,6148649	0,6486486
9	0,6013514	0,6283784	0,6081081	0,6216216	0,6351351	0,6689189	0,6283784
10	0,6824324	0,6689189	0,6351351	0,6013514	0,6418919	0,6756757	0,5945946
Media	**0,64864865**	**0,66351351**	**0,65405405**	**0,65675676**	**0,64662162**	**0,65202703**	**0,64932432**

Tabla 19: Resultados de la prueba 3 de Yeast

El resumen de las pruebas muestra los siguientes resultados (Tabla 20):

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
Media	0,65788288	0,66711712	0,66081081	0,66689189	0,66148649	0,66238739	0,65968468

Tabla 20: Resumen de resultados de las pruebas de Yeast

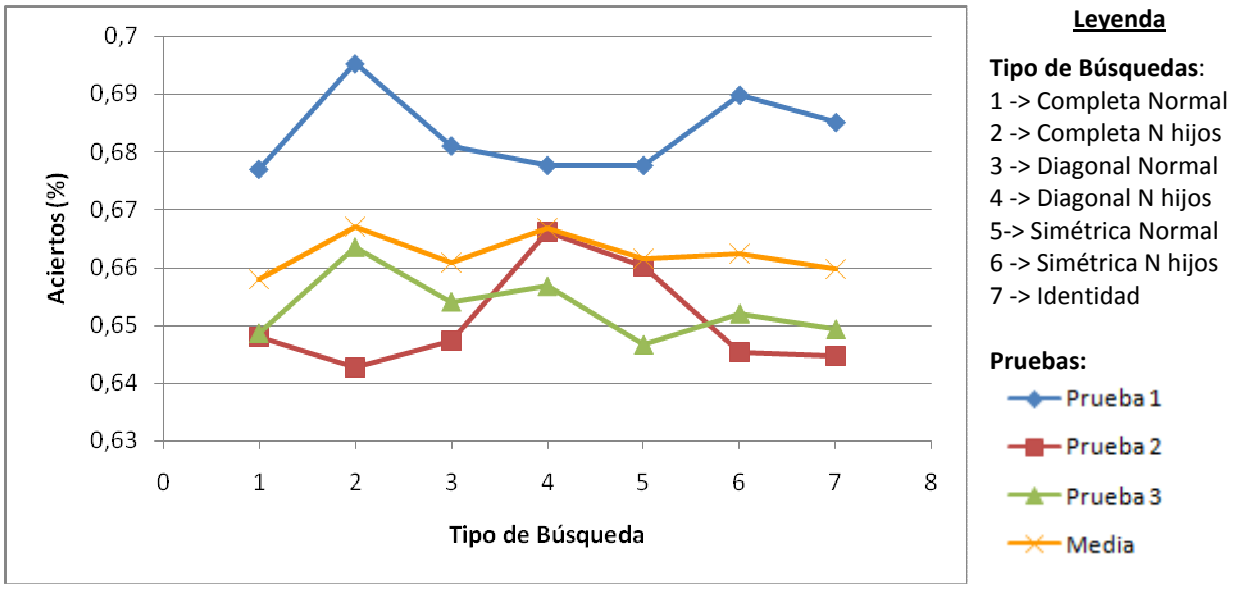


Imagen 44: Representación gráfica de las pruebas de Yeast

Los resultados de las pruebas realizadas a este dominio denotan una cierta superioridad de las ejecuciones con el modo N hijos frente a ejecuciones normales de los diferentes tipos de búsquedas.

La prueba que mejor porcentaje de aciertos ha obtenido es la Prueba 1, con la matriz completa con ejecución N hijos. Este caso supera en aproximadamente un 1% al caso base establecido por la matriz identidad en dicha prueba.

La matriz diagonal con modo de ejecución N hijos ha obtenido en media unos resultados muy próximos a los del mejor caso, variando en apenas un 0,02% sus resultados con éste.

Con estos resultados, se puede afirmar que el tipo de matriz que mejor porcentaje de acierto ha logrado durante la clasificación de las instancias de test ha sido matriz completa en su ejecución N hijos. Se puede concluir con una tendencia de este tipo de matriz a mejorar los porcentajes de aciertos para este dominio, respecto a la matriz identidad.

4.6 Dominio EEG

El dominio de datos EEG contiene instancias basadas en el registro de la actividad bioeléctrica cerebral. Este registro se basa en una exploración neurofisiológica y proviene de la electroencefalografía (EEG). Se pueden tomar datos en diferentes situaciones o condiciones, como por ejemplo: en reposo, durante el sueño, realizando diferentes actividades, etc.

A continuación se muestra una imagen en la que se indica la posición de los electrodos en la cabeza:

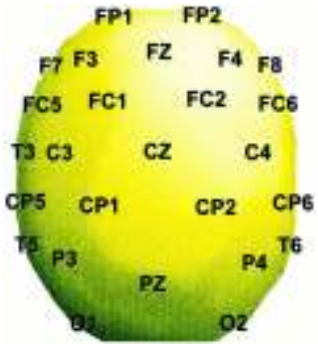


Imagen 45: Posición de los electrodos en EEG

Los datos utilizados han sido tomados mientras una persona manejaba un ordenador.

Los datos usados para realizar experimentaciones con el sistema desarrollado se encontraban separados en dos conjuntos, uno para entrenamiento y otro para test.

Conjunto datos	Instancias totales	Atributos	Clases
Entrenamiento	**6395**	**6**	**2**
Test	**2075**	**6**	**2**

Tabla 21: Características del dominio EEG

Nota: Los atributos no incluyen el atributo que describe la clase de la instancia.

Los atributos que componen las instancias del dominio EEG son:

1. C3-2. Actividad eléctrica espontánea en la corteza cerebral medida por el electrodo C3 en la banda de frecuencia 2.
2. C3-3. Actividad eléctrica espontánea en la corteza cerebral medida por el electrodo C3 en la banda de frecuencia 3.
3. C4-2. Actividad eléctrica espontánea en la corteza cerebral medida por el electrodo C4 en la banda de frecuencia 2.

4. C4-3. Actividad eléctrica espontánea en la corteza cerebral medida por el electrodo C4 en la banda de frecuencia 3.
5. CP1-2. Actividad eléctrica espontánea en la corteza cerebral medida por el electrodo CP1 en la banda de frecuencia 2.
6. CP1-3. Actividad eléctrica espontánea en la corteza cerebral medida por el electrodo CP1 en la banda de frecuencia 3.
7. Variable de la clase:
    - 2. Esta clase indica imaginación del movimiento de la mano izquierda.
    - 3. esta clase indica imaginación del movimiento de la mano derecha.

Las pruebas realizadas para comprobar el funcionamiento del algoritmo desarrollado en el dominio EEG han sido realizadas con la siguiente configuración:

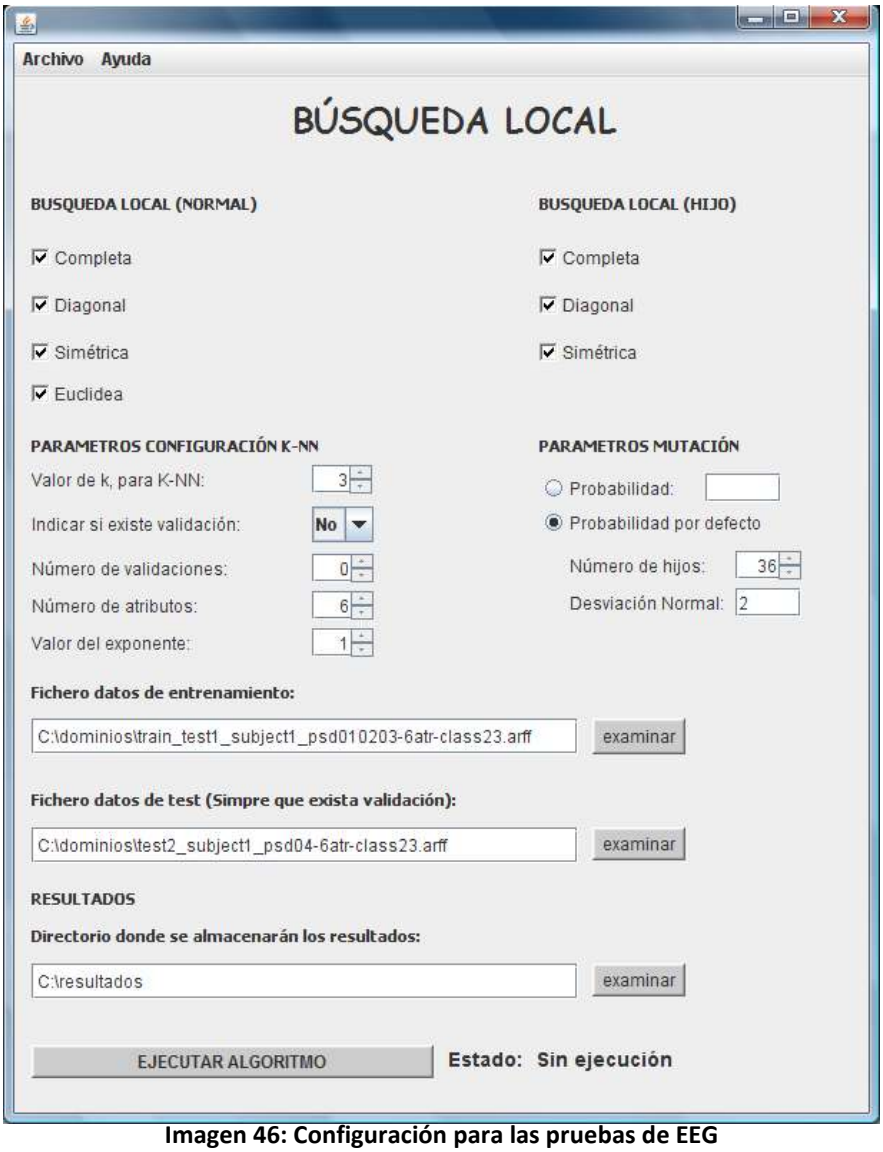


Imagen 46: Configuración para las pruebas de EEG

Debido a las características especiales de este dominio no se utilizará validación cruzada. Esto se debe a que los datos obtenidos para los conjuntos de entrenamiento y test han sido obtenidos en diferentes sesiones. No conviene mezclar las sesiones, puesto que el objetivo de este dominio es realizar predicciones en sesiones distintas. Por ello, se ha usará un conjunto para entrenamiento y otro para test.

Prueba 1

Resultados de la primera prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
1	0,8032946	0,7771318	0,7994186	0,7965116	0,7868217	0,7873062	0,7829457

Tabla 22: Resultados de la prueba 1 de EEG

Prueba 2

Resultados de la segunda prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
1	0,7877907	0,7868217	0,7955426	0,7974806	0,7931202	0,7843992	0,7829457

Tabla 23: Resultados de la prueba 2 de EEG

Prueba 3

Resultados de la tercera prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
1	0,7906977	0,7892442	0,7965116	0,7950581	0,7921512	0,7882752	0,7829457

Tabla 24: Resultados de la prueba 3 de EEG

El resumen de las pruebas muestra los siguientes resultados (Tabla 25):

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
Media	0,7939276	0,7843992	0,7971576	0,7963501	0,7906977	0,7866602	0,7829457

Tabla 25: Resumen de resultados de las pruebas de EEG

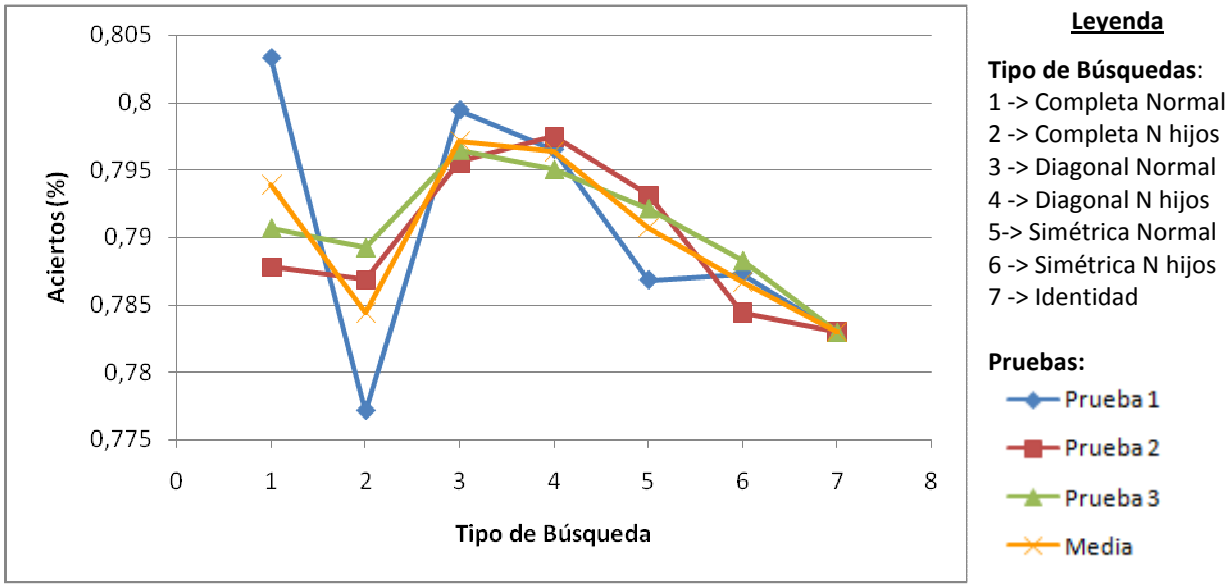


Imagen 47: Representación gráfica de las pruebas de EEG

El resumen de resultados muestra como el tipo de matriz que mejor ha funcionado en este dominio, y que por lo tanto mejor porcentaje (en media) de aciertos en test ha conseguido es la matriz diagonal. En las tres ejecuciones realizadas, y con resultados muy similares, se consigue mejorar en aproximadamente un 1,5% el caso de la matriz identidad.

Por otro lado, la matriz completa en su ejecución normal también ha logrado buenos resultados, mejorando en más de un 1% el porcentaje de aciertos en test de la matriz identidad.

Observando la imagen 47 se puede contemplar un patrón de aciertos muy similar para los diferentes tipos de matrices y para todas las pruebas.

Por último, comentar que los resultados extraídos de la ejecución del algoritmo desarrollado en el dominio de EEG desvelan que la matriz que mejores porcentajes de aciertos en test ha logrado ha sido la matriz diagonal. Lo ha logrado en su modo de ejecución normal.

4.7 Dominios Sintéticos

Este punto describirá la experimentación realizada con los datos pertenecientes a un conjunto de dominios que han sido generados de forma sintética específicamente para probar el método desarrollado. En estos dominios se podrá comprobar la potencia del algoritmo implementado ya que se conoce qué tipo de matriz permitirá mejorar los porcentajes de aciertos.

4.7.1 Rectas Ruido Horizontales

Dominio con datos generados de forma sintética que representan dos rectas horizontales con ruido cuyos datos pertenecientes a dos clases diferentes. En este dominio, un dato tendrá como vecino más cercano a un dato de la clase contraria.

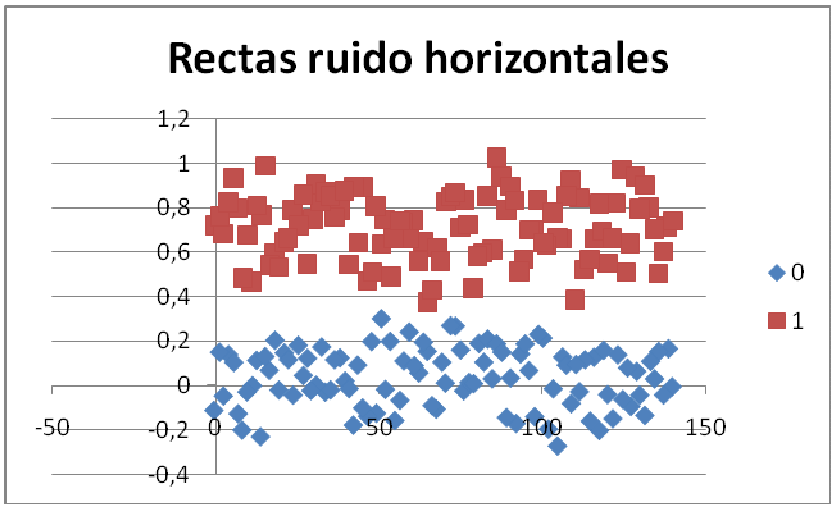


Imagen 48: Distribución de los datos del dominio Rectas ruido horizontales

Esta imagen puede ser engañosa, debido a las escalas de los ejes. A continuación se muestra un zoom de una parte de estos datos para observar mejor la distribución de los mismos.

Imagen 49: Zoom de la distribución de datos del dominio Rectas Ruido Horizontales

Se puede ver que cada punto tiene como vecino más cercano a un punto perteneciente a la clase contraria. Por lo tanto, el algoritmo K-NN con k=1 (solamente el vecino más cercano) obtendrá un porcentaje de aciertos cercano a 0.

Vemos en este caso que un escalado de los ejes, es decir un matriz diagonal, puede hacer que el porcentaje de aciertos se acerque al 100%.

Información acerca de los datos utilizados:

Instancias totales	Atributos	Clases
200	**2**	**2**

Tabla 26: Características del dominio Rectas Ruido Horizontales

Nota: Los atributos no incluyen el atributo que describe la clase de la instancia.

Los atributos que componen las instancias del dominio Rectas Ruido Horizontales son:

1. Coordenada x (número real).
2. Coordenada y (número real).
3. Variable de la clase:
   - 0
   - 1

Las pruebas realizadas para comprobar el funcionamiento del algoritmo desarrollado en el dominio Rectas Ruido Horizontales han sido realizadas con la siguiente configuración:

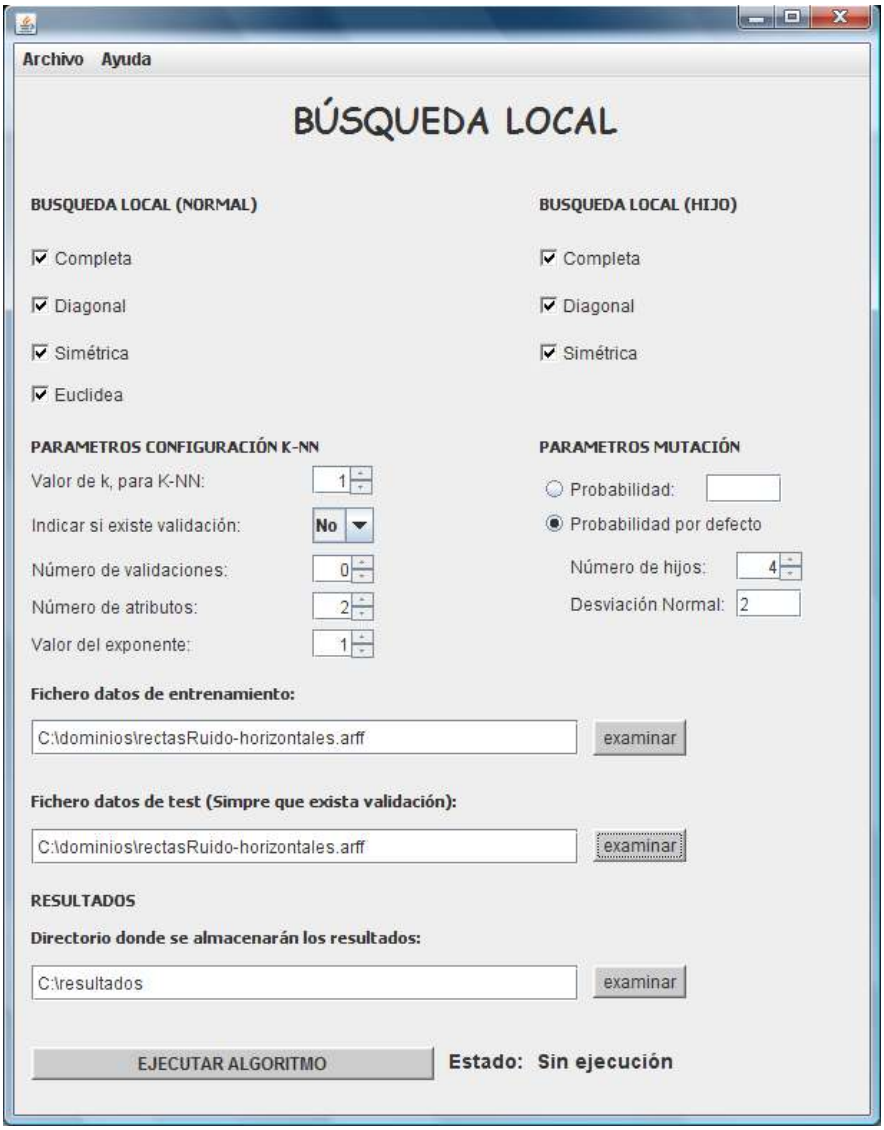


Imagen 50: Configuración para las pruebas de Rectas Ruido Horizontales

Destacar que se utilizará el mismo fichero de datos para entrenamiento y para test, debido a que si se realizaran validaciones del conjunto este se "rompería" y podría suceder que un punto tuviera como vecino más cercano a otro de la misma clase. Por ello, el resultado obtenido en test no será el resultado de esta prueba, ya que será siempre un porcentaje de acierto del 100% al encontrarse el propio dato en los dos conjuntos. Se obtendrá como resultado el valor del fitness obtenido en la fase de entrenamiento.

Prueba 1

Resultados de la primera prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
1	0,985	0,015	0,015	1	0,96	0,015	0,015

Tabla 27: Resultados de la prueba 1 de Rectas Ruido Horizontales

Prueba 2

Resultados de la segunda prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
1	1	0,97	0,015	0,975	0,99	0,975	0,015

Tabla 28: Resultados de la prueba 2 de Rectas Ruido Horizontales

Prueba 3

Resultados de la tercera prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
1	0,98	0,7	0,97	0,825	0,975	0,035	0,015

Tabla 29: Resultados de la prueba 3 de Rectas Ruido Horizontales

El resumen de las pruebas muestra los siguientes resultados (Tabla 30):

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
Media	0,9883333	0,56166667	0,3333333	0,9333333	0,975	0,3416667	0,015

Tabla 30: Resumen de resultados de las pruebas de Rectas Ruido Horizontales

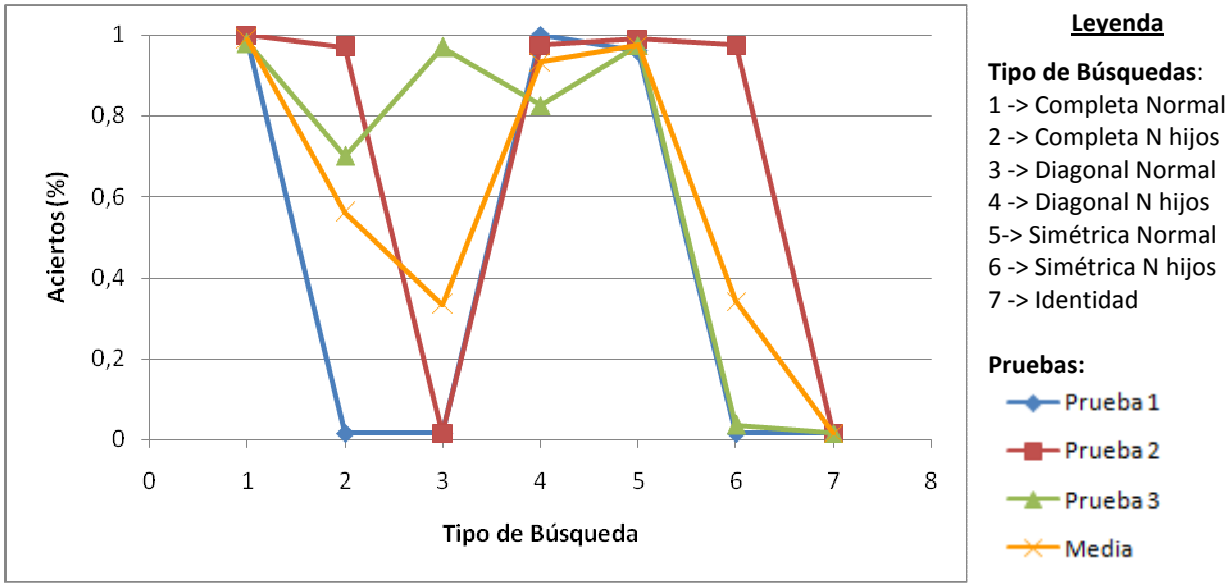


Imagen 51: Representación gráfica de las pruebas de Rectas Ruido Horizontales

Los resultados obtenidos empleando la media de las tres pruebas denotan cómo las matrices completas y simétricas han conseguido muy buenos resultados en su ejecución normal. Los mejores porcentajes de acierto, en media, han sido los siguientes: 98,88% para la matriz completa y 97,5% para la matriz simétrica.

Por el contrario, la matriz diagonal ha obtenido mejores resultados para la ejecución con N hijos, alcanzando un 93,3% de aciertos.

Todas las pruebas han demostrado la eficiencia del algoritmo desarrollado, ya que han superado con creces el 1,5% de aciertos de la matriz identidad. Sin embargo hay que destacar también la posibilidad de que algunos de los algoritmos se estanquen en mínimos locales en algunas de las pruebas.

La matriz que mejores resultados ha obtenido en media ha sido la matriz completa en su ejecución con normal.

Finalizar las conclusiones de esta prueba con la sorpresa de que la matriz diagonal, para su ejecución normal, no ha logrado los resultados esperados. Estos resultados se deben probablemente a que la evolución de la matriz diagonal únicamente genera 2 hijos, con la consiguiente probabilidad de que ninguno mejore los resultados de su progenitor.

A continuación se muestra el resultado obtenido con una de las matrices. Servirá para verificar el correcto funcionamiento de la evolución de las matrices.

El resultado muestra la proyección de los datos respecto a la matriz completa obtenida como mejor resultado en la prueba 1. La matriz obtenida mediante el algoritmo de búsqueda local fue la siguiente:

$$\begin{bmatrix} 1 & 0 \\ 1{,}70370877 & 5{,}84406189 \end{bmatrix}$$

Imagen 52: Matriz obtenida para el dominio Rectas ruido horizontales

En la siguiente imagen se muestran los datos iniciales y los datos proyectados del dominio que permitirán realizar comparaciones posteriormente.

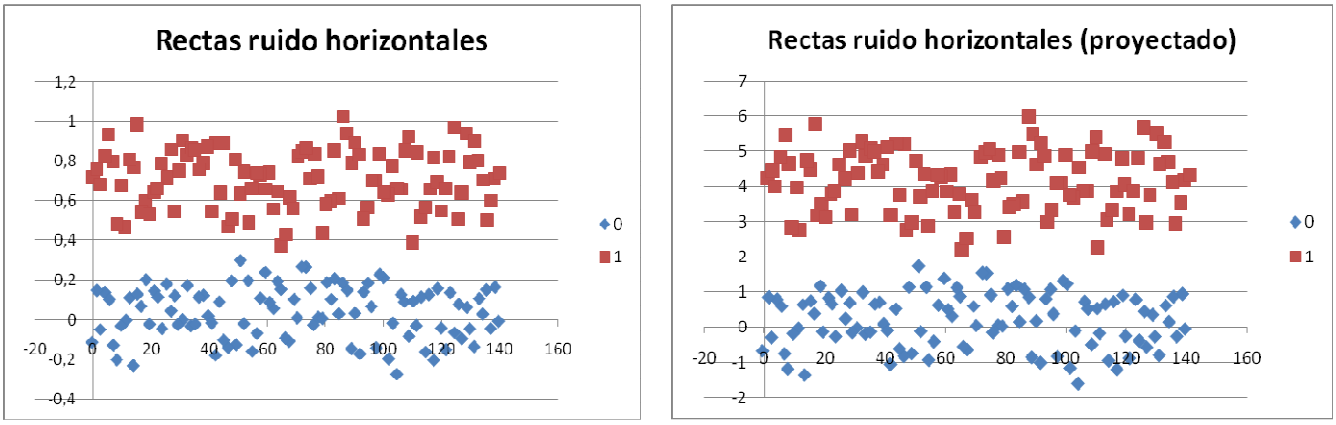


Imagen 53: Comparativa de datos originales y proyectados para Rectas ruido horizontales

Observando las gráficas se demuestra como la matriz ha realizado el escalado del eje de ordenadas.

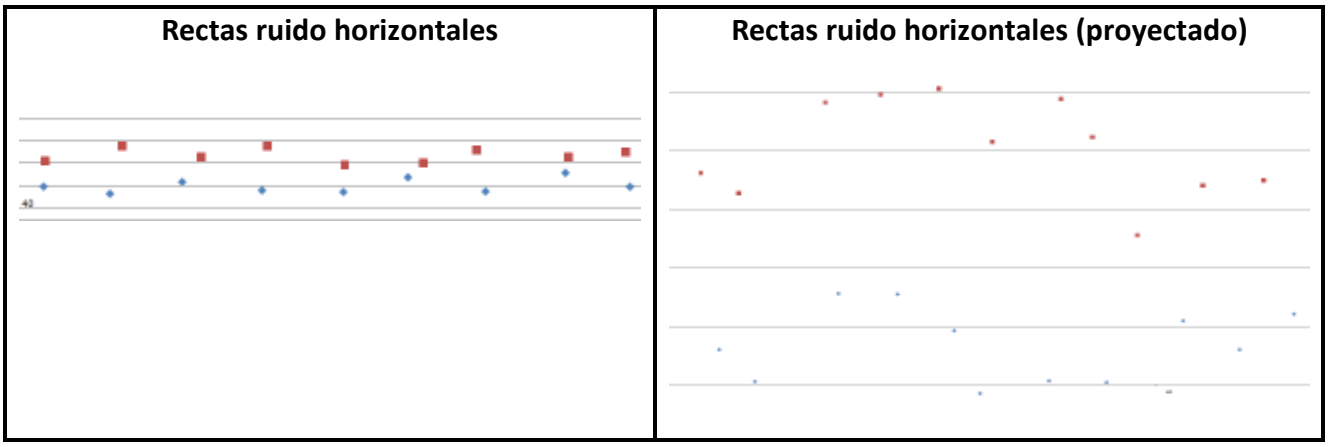


Imagen 54: Zoom de la comparativa de datos originales y proyectados para Rectas ruido horizontales

Si se observa la imagen con zoom, se verifica que tras proyectar los datos el punto más cercano a cualquier punto casi siempre es uno de la misma clase. Aumentando el porcentaje de éxito hasta casi el 100%.

4.7.2 Rectas Ruido 45

Dominio con datos generados de forma sintética que representan dos rectas formando un ángulo de 45 grados con el eje de abscisas.

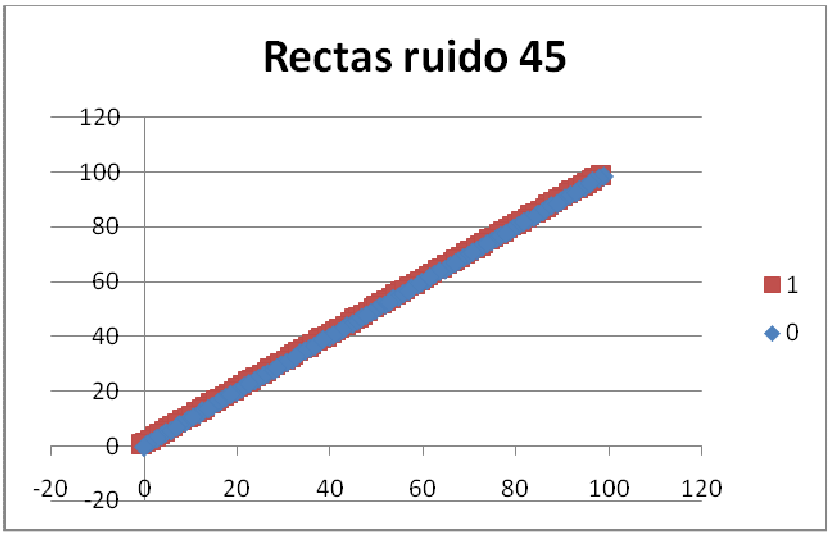


Imagen 55: Distribución de los datos del dominio Rectas ruido 45

En esta imagen se puede apreciar como los datos de ambas clases se encuentran muy juntos. Para obtener una mejor visión del conjunto de datos utilizados en este dominio se ha realizado un zoom de una sección de los datos:

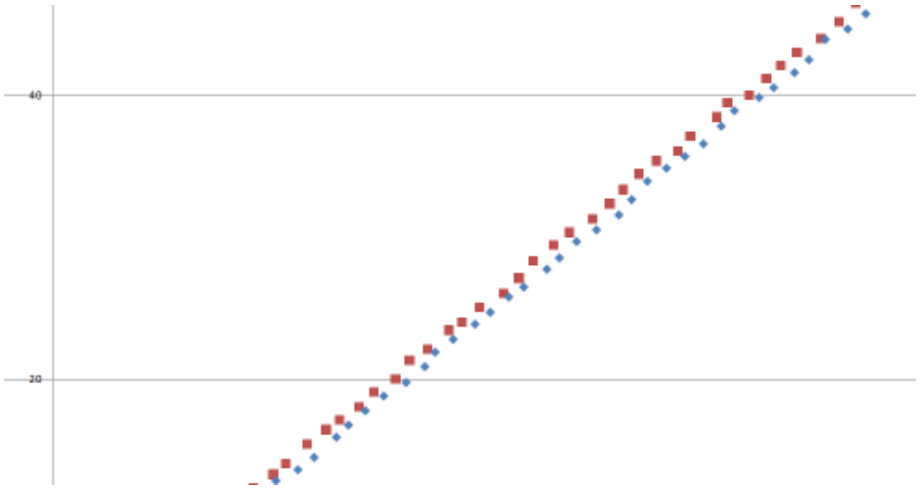


Imagen 56: Zoom de la distribución de datos del dominio Rectas Ruido 45

El zoom realizado muestra como los datos de cualquiera de las clases tiene como único vecino más cercano a un dato de la clase opuesta. Por esta causa, cuando posteriormente ejecutemos el algoritmo se podrá observar que usando la matriz identidad el porcentaje de acierto es prácticamente 0.

Se observa que un escalado de los ejes (obtenido con la matriz diagonal) no será suficiente para obtener buenos resultados. Se necesita además una rotación que solamente pueden conseguir las matrices completa o simétrica.

Información acerca de los datos utilizados:

Instancias totales	Atributos	Clases
200	**2**	**2**

Tabla 31: Características del dominio Rectas Ruido 45

Nota: Los atributos no incluyen el atributo que describe la clase de la instancia.

Los atributos que componen las instancias del dominio Rectas Ruido 45 son:

1. Coordenada x (número real).
2. Coordenada y (número real).
3. Variable de la clase:
   - 0
   - 1

Las pruebas realizadas para comprobar el funcionamiento del algoritmo desarrollado en el dominio Rectas Ruido 45 han sido realizadas con la siguiente configuración:

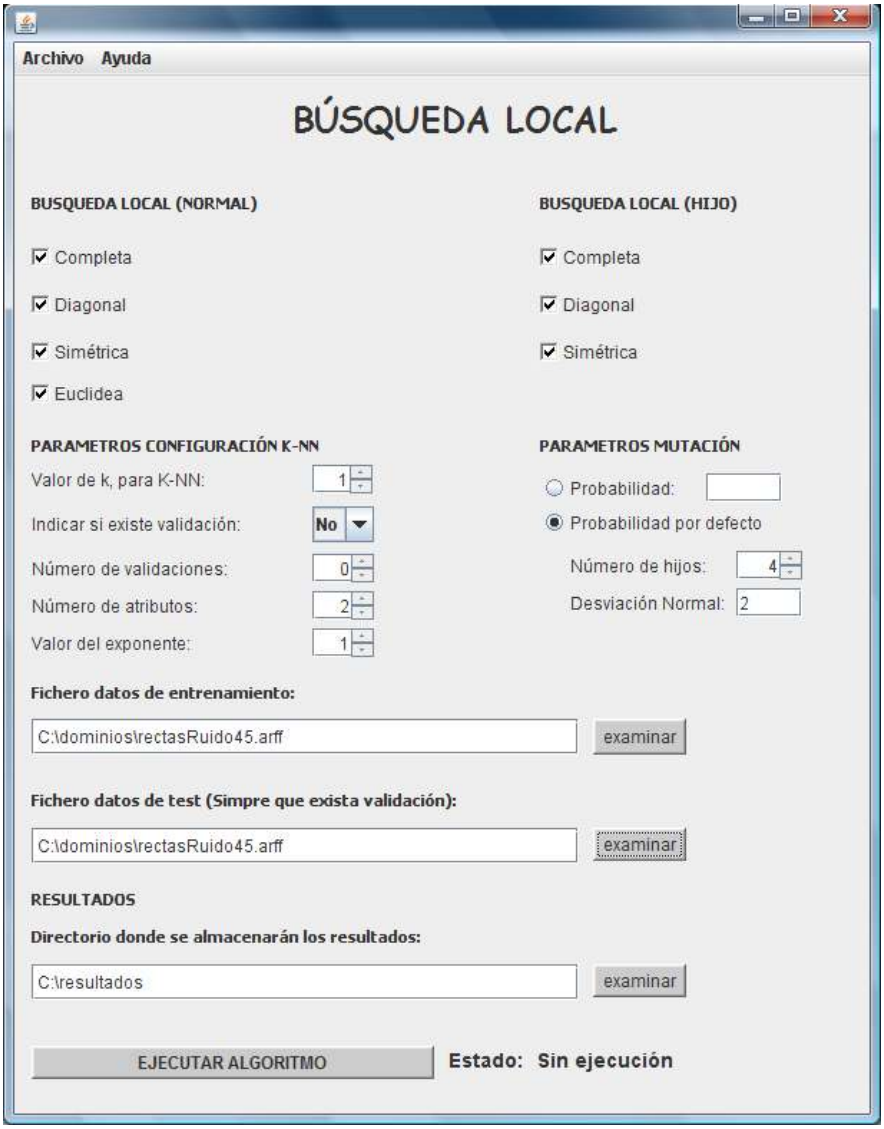


Imagen 57: Configuración para las pruebas de Rectas Ruido 45

La configuración para esta prueba es similar a la del primer caso de dominio sintético. Se utilizará el mismo fichero de datos para entrenamiento y test. El valor de k para el algoritmo K-NN es de 1, provocando así que el vecino más cercano casi siempre sea de la clase opuesta.

Prueba 1

Resultados de la primera prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**
1	0,71	0,88	0,02	0,02	0,12	0,075	0,01

Tabla 32: Resultados de la prueba 1 de Rectas Ruido 45

Prueba 2

Resultados de la segunda prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**
1	0,7	0,01	0,01	0,015	0,065	0,99	0,01

Tabla 33: Resultados de la prueba 2 de Rectas Ruido 45

Prueba 3

Resultados de la tercera prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**	**N hijos**	**Normal**
1	0,995	0,58	0,03	0,01	0,01	0,885	0,01

Tabla 34: Resultados de la prueba 3 de Rectas Ruido 45

El resumen de las pruebas muestra los siguientes resultados (Tabla 35):

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
Media	0,8016667	0,49	0,02	0,015	0,065	0,65	0,01

Tabla 35: Resumen de resultados de las pruebas de Rectas Ruido 45

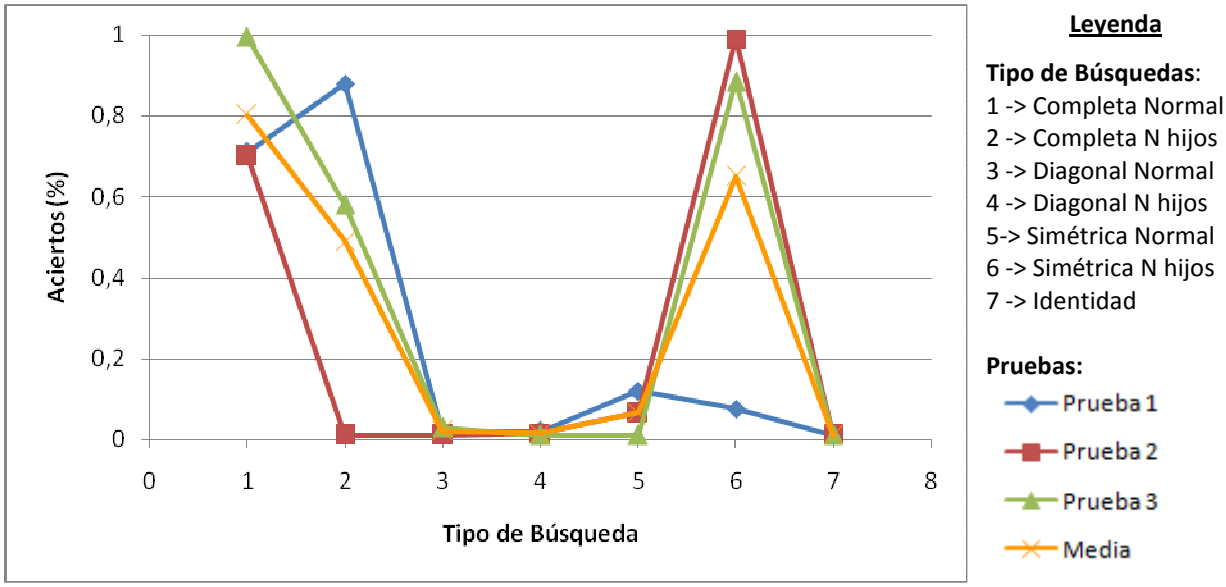


Imagen 58: Representación gráfica de las pruebas de Rectas Ruido 45

Los resúmenes de las pruebas muestran los resultados que se podían esperar según han ido transcurriendo las pruebas, la matriz completa en su ejecución normal ha conseguido en media un porcentaje de acierto del 80,17%.

La matriz simétrica también ha logrado evolucionar hasta conseguir un 65% de media para su ejecución normal.

La matriz diagonal, como era esperado, no ha conseguido apenas mejorar los resultados obtenidos por la matriz identidad, que tal y como se comentó durante las pruebas solamente acertaba un 1% de los datos. Esto se debe a que se necesitaba un escalado de los ejes unido a un giro de los mismos, situación que la matriz diagonal no puede conseguir en ningún caso.

Se puede afirmar que los resultados obtenidos en las pruebas realizadas a este dominio representan las expectativas que se tenían antes de ellas. Se ha ratificado que la matriz diagonal no puede obtener buenos porcentajes de éxito en este dominio, y que por el contrario las matrices completa y simétrica sí son capaces de obtener buenos resultados.

La matriz completa ha sido la que ha logrado mejor porcentaje de aciertos. Los ha logrado en su ejecución normal.

Al disponer este dominio de dos atributos, a continuación se representará la proyección de los datos realizada con una de las matrices obtenidas tras evolucionar mediante el algoritmo de búsqueda local.

La matriz que se utilizará para proyectar los datos será la matriz completa en su ejecución normal. Corresponde a la prueba 3, y sus elementos son los siguientes:

-3,57403623	-0,43002753
4,22462251	1

Imagen 59: Matriz obtenida para el dominio Rectas ruido 45

En la siguiente imagen se muestran los datos iniciales y los datos proyectados del dominio que permitirán realizar comparaciones posteriormente.

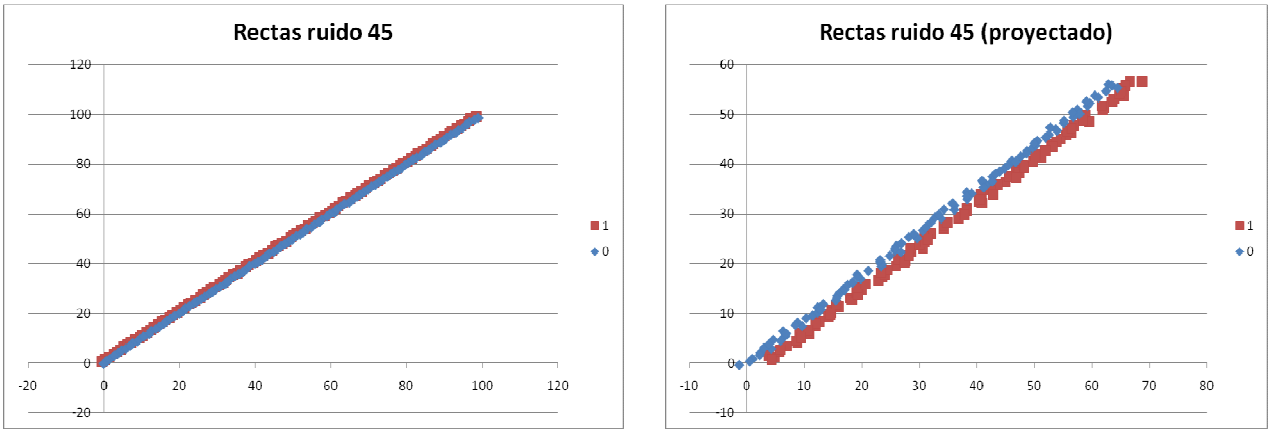


Imagen 60: Comparativa datos originales y proyectados para Rectas ruido 45

En las gráficas anteriores se puede observar como al proyectar los datos se han separado manteniendo dos rectas, una para cada clase.

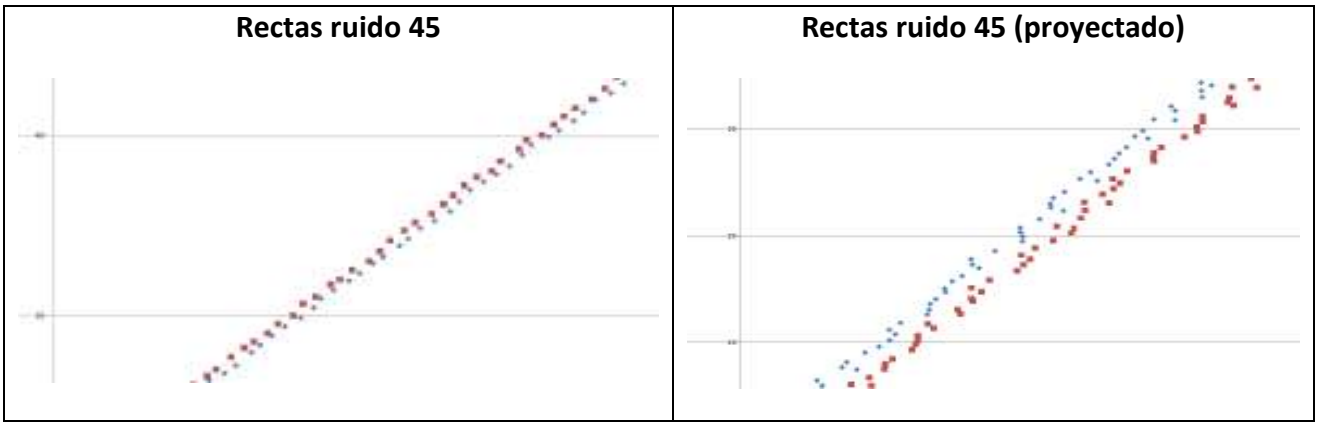


Imagen 61: Zoom de la comparativa datos originales y proyectados para Rectas ruido 45

Si se observa el zoom de la comparativa se puede ver como los puntos de una clase tienen como vecino más cercano a un punto de la misma clase, para el casi el 100% de los datos.

4.7.3 Aleatorio

El dominio aleatorio contiene instancias generadas sintéticamente. Las instancias de este dominio contienen 4 atributos. Dos de ellos representan una recta diagonal, y los otros solamente sirven para generar ruido.

Información acerca de los datos utilizados:

Instancias totales	Atributos	Clases
300	**4**	**2**

Tabla 36: Características del dominio Aleatorio

Nota: Los atributos no incluyen el atributo que describe la clase de la instancia.

Los atributos que componen las instancias del dominio Aleatorio son:

1. Coordenada 1 (número real).
2. Coordenada 2 (número real).
3. Coordenada 3 (número real).
4. Coordenada 4 (número real).
5. Variable de la clase:
   - 0
   - 1

Las pruebas realizadas para comprobar el funcionamiento del algoritmo desarrollado en el dominio Aleatorio han sido realizadas con la siguiente configuración:

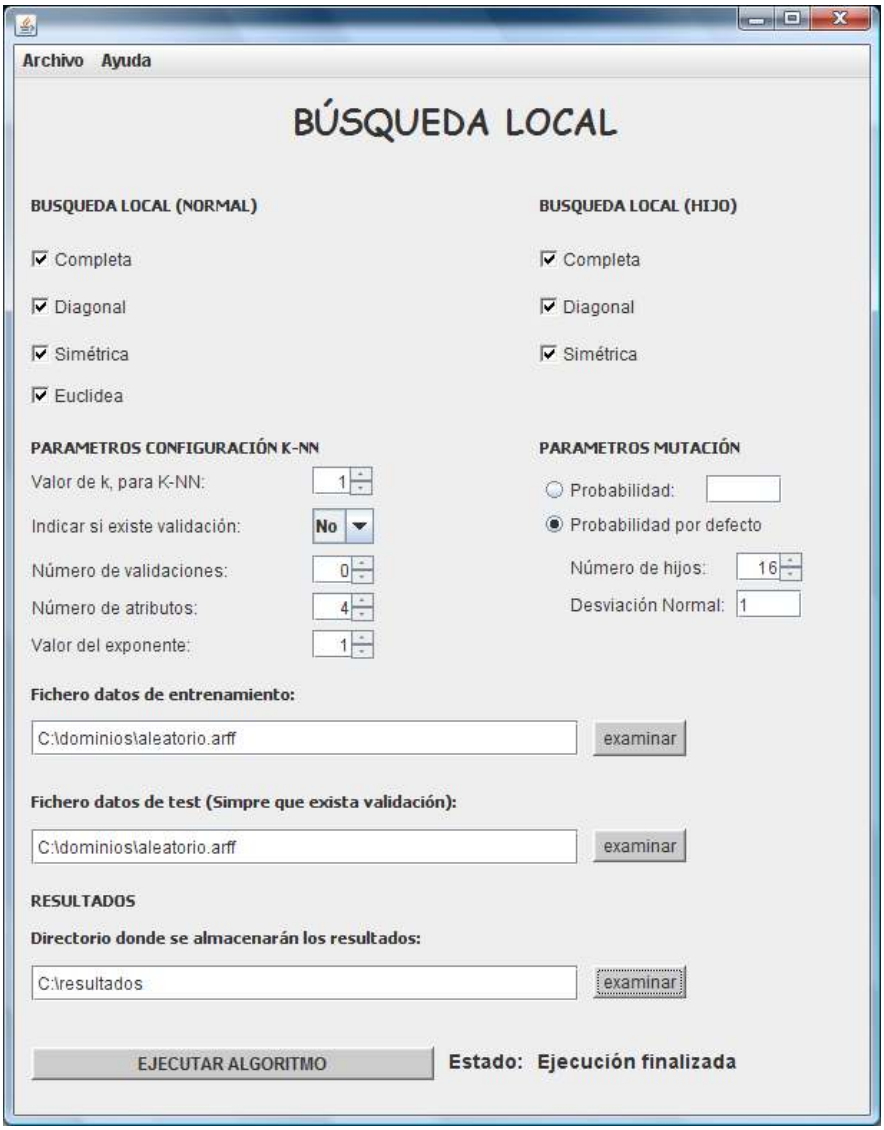


Imagen 62: Configuración para las pruebas de Aleatorio

Al igual que el resto de dominios sintéticos analizados hasta ahora, en este dominio también se utilizará el mismo fichero para entrenamiento y test. Recordar que el resultado final obtenido saldrá de la fase de entrenamiento.

Prueba 1

Resultados de la primera prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
1	0,98666667	0,92666667	0,92666667	0,97333333	0,92666667	0,98	0,92666667

Tabla 37: Resultados de la prueba 1 de Aleatorio

Prueba 2

Resultados de la segunda prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
1	0,98	0,97666667	0,94333333	0,95666667	0,98	0,99	0,92666667

Tabla 38: Resultados de la prueba 2 de Aleatorio

Prueba 3

Resultados de la tercera prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
1	0,96666667	0,96666667	0,92666667	0,95666667	0,95666667	0,95	0,92666667

Tabla 39: Resultados de la prueba 3 de Aleatorio

El resumen de las pruebas muestra los siguientes resultados (Tabla 40):

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
Media	0,97777778	0,95666667	0,93222222	0,96222222	0,95444444	0,97333333	0,92666667

Tabla 40: Resumen de resultados de las pruebas de Aleatorio

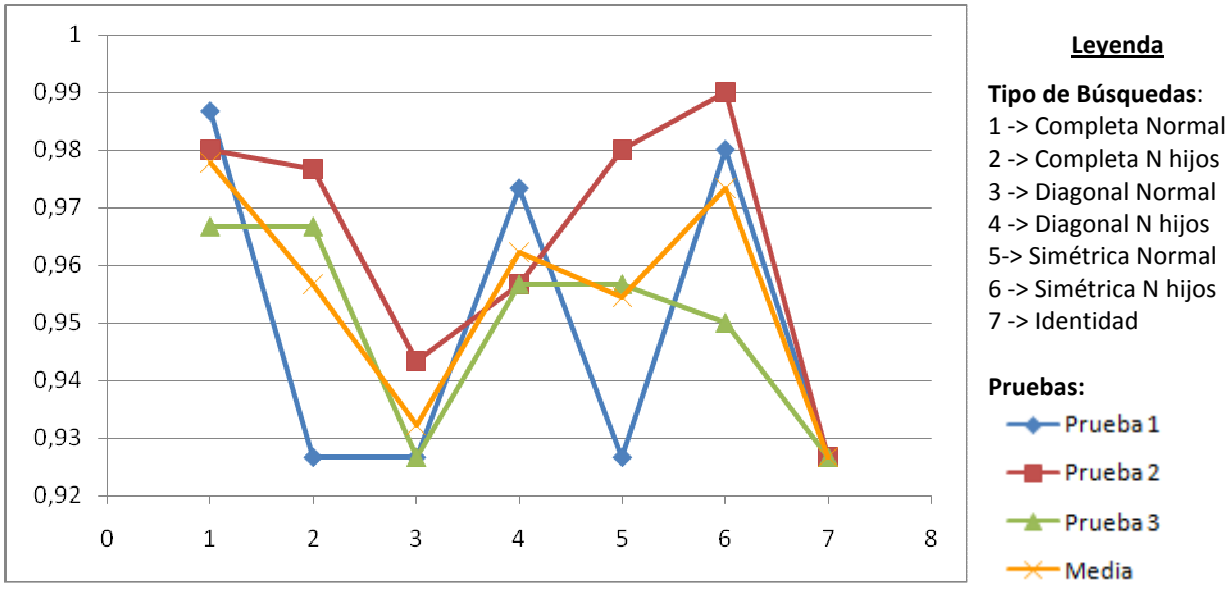


Imagen 63: Representación gráfica de las pruebas de Aleatorio

Estudiando los resultados en media para cada tipo de matriz se observa un patrón generalizado en el gráfico bastante similar para todas las matrices y ambos modos de ejecución en las pruebas.

Se observa como la matriz completa funciona correctamente alcanzando casi un 98% de acierto frente al 92,67% de la matriz identidad.

Los resultados obtenidos por las matrices diagonal y simétrica para su ejecución N hijos también demuestran la favorable evolución de estas matrices con el algoritmo desarrollado. Por tanto, se puede destacar la tendencia de las matrices completa (ejecución normal), diagonal (ejecución N hijos) y simétrica (ejecución N hijos) a mejorar los resultados obtenidos por la matriz identidad.

La conclusión alcanzada con las 3 pruebas realizadas identifica a la matriz completa en su ejecución con normal como la que mejor clasifica los datos de este dominio.

4.7.4 Aleatorio Girado

Dominio con datos generados a partir del dominio Aleatorio (*ver punto anterior*). Los datos de este nuevo dominio han sido generados mediante la multiplicación de los datos del dominio Aleatorio por una matriz 4D. Esta multiplicación provoca una proyección de los datos.

Información acerca de los datos utilizados:

Instancias totales	Atributos	Clases
300	**4**	**2**

Tabla 41 : Características del dominio Aleatorio Girado

Nota: Los atributos no incluyen el atributo que describe la clase de la instancia.

Los atributos que componen las instancias del dominio Aleatorio Girado son:

1. Coordenada 1 (número real).
2. Coordenada 2 (número real).
3. Coordenada 3 (número real).
4. Coordenada 4 (número real).
5. Variable de la clase:
   - 0
   - 1

Las pruebas realizadas para comprobar el funcionamiento del algoritmo desarrollado en el dominio Aleatorio Girado han sido realizadas con la siguiente configuración:

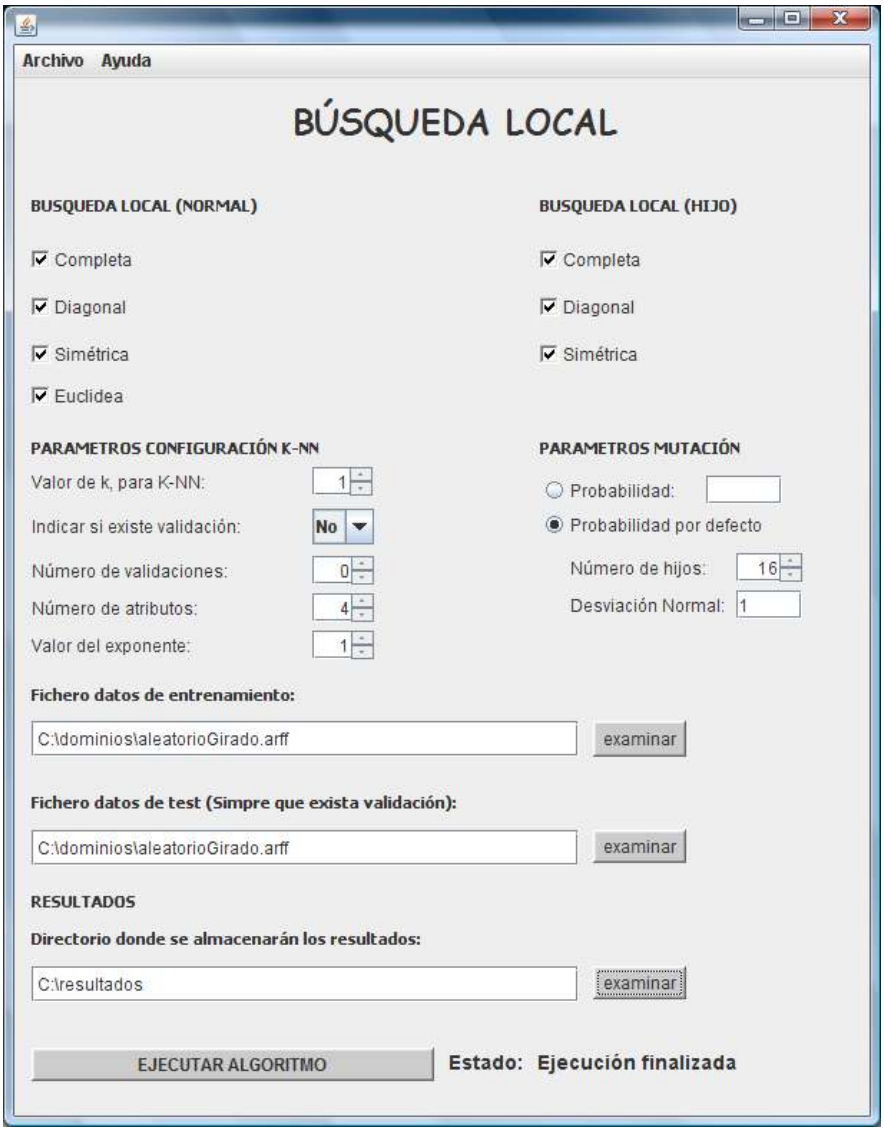


Imagen 64: Configuración para las pruebas de Aleatorio Girado

Nuevamente se emplea el mismo fichero para datos y para test. El resultado que utilizaremos para posteriormente realizar comparaciones y obtener un índice de eficacia será nuevamente el porcentaje de aciertos durante el entrenamiento.

Prueba 1

Resultados de la primera prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
1	0,95	0,94666667	0,90333333	0,91	0,94	0,94	0,88666667

Tabla 42: Resultados de la prueba 1 de Aleatorio Girado

Prueba 2

Resultados de la segunda prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
1	0,94	0,94666667	0,90333333	0,90666667	0,92	0,94	0,88666667

Tabla 43: Resultados de la prueba 2 de Aleatorio Girado

Prueba 3

Resultados de la tercera prueba realizada:

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
Validación test	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
1	0,95333333	0,94333333	0,90666667	0,92333333	0,96666667	0,93666667	0,88666667

Tabla 44: Resultados de la prueba 3 de Aleatorio Girado

El resumen de las pruebas muestra los siguientes resultados (Tabla 45):

	COMPLETA		DIAGONAL		SIMETRICA		IDENTIDAD
	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
Media	0,94777778	0,94555556	0,90444444	0,91333333	0,94222222	0,93888889	0,88666667

Tabla 45: Resumen de resultados de las pruebas de Aleatorio Girado

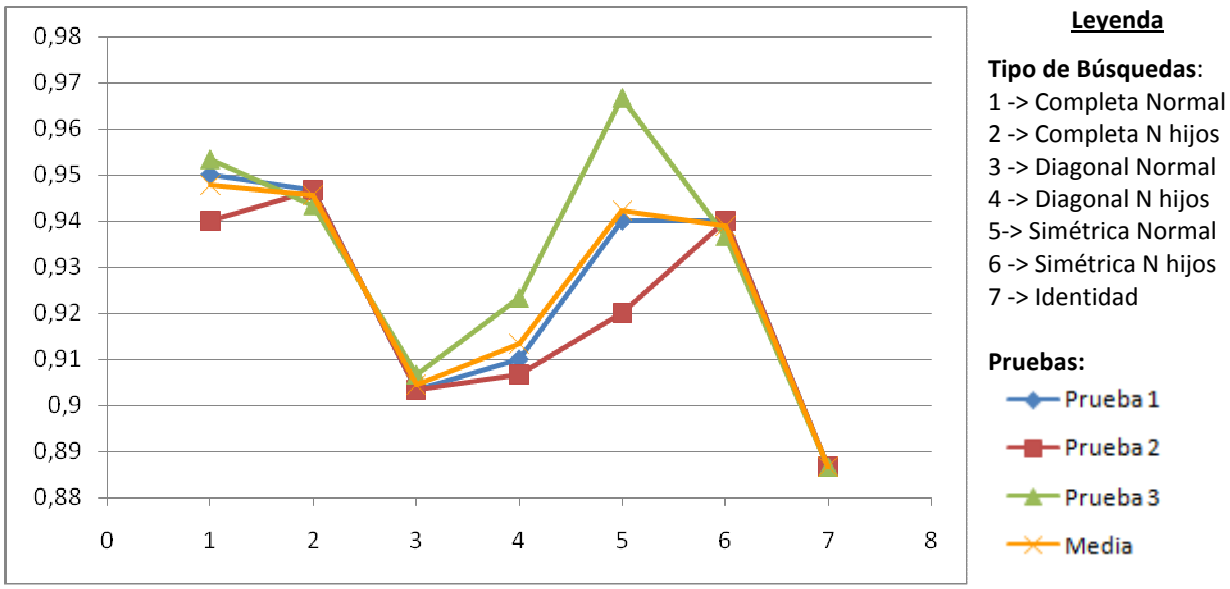


Imagen 65: Representación gráfica de las pruebas de Aleatorio Girado

Los datos obtenidos mediante la media de las 3 pruebas para este dominio también denotan un patrón similar en los resultados obtenidos por cada tipo de búsqueda. Este hecho también se produjo en el domino anterior, del cual proceden los datos del actual dominio.

Al igual que el dominio anterior, los tipos de matrices que mejor funcionan son la matriz completa y la matriz simétrica. Los resultados obtenidos por la matriz diagonal en ambos dominios son inferiores a los de las matrices anteriormente mencionadas.

Observando la imagen 65 se puede ver la tendencia de la matrices completa y simétrica en su ejecución normal a mejorar los resultados de la matriz identidad.

El mejor tipo de matriz alcanza un porcentaje en media de 94,78% de aciertos en media frente al 88,67% de la matriz identidad. En este dominio, por lo tanto, el tipo de matriz que mejores porcentajes de éxitos ha logrado es la matriz completa en su modo de ejecución normal.

Por último se presenta una tabla resumen que recoge los resultados obtenidos en media para cada tipo de matriz en todos los dominios utilizados en la fase de experimentación.

	COMPLETA		DIAGONAL		SIMETRICA		EUCLIDEA
	Normal	N hijos	Normal	N hijos	Normal	N hijos	Normal
Ripley	**0,9312**	0,92853333	0,92906667	0,92986667	0,93013333	0,9296	0,92986667
Diabetes	0,79385965	0,79254386	**0,78947368**	0,79824561	0,7877193	0,77938596	0,77192982
Iris	0,96444444	0,97555556	0,96666667	**0,98**	0,96444444	0,97555556	0,96666667
Yeast	0,65788288	**0,66711712**	0,66081081	0,66689189	0,66148649	0,66238739	0,65968468
EEG	0,79392765	0,78439922	**0,79715762**	0,79635013	0,79069767	0,78666021	0,78294574
Rectas ruido horizontales	**0,98833333**	0,56166667	0,33333333	0,93333333	0,975	0,34166667	0,015
Rectas ruido 45	**0,80166667**	0,49	0,02	0,015	0,065	0,65	0,01
Aleatorio	**0,97777778**	0,95666667	0,93222222	0,96222222	0,95444444	0,97333333	0,92666667
Aleatorio girado	**0,94777778**	0,94555556	0,90444444	0,91333333	0,94222222	0,93888889	0,88666667

Tabla 46: Resumen de la experimentación

Observando la tabla que muestra el resumen de la experimentación realizadas para todos los dominios (Tabla 46) se denota una clara tendencia a que la matriz completa es el tipo de matriz que mejores resultados puede obtener.

Por el contrario, la matriz simétrica no ha conseguido imponerse en media en ninguno de los dominios utilizados para esta fase de experimentación, a pesar de ello, los resultados obtenidos con este tipo de matriz son satisfactorios.

Capítulo 5

Conclusiones

El objetivo de este capítulo es ofrecer al lector las conclusiones finales que se extraen del proyecto realizado. Además, se ofrecen algunas posibles líneas futuras de investigación para el proyecto desarrollado.

5.1 Conclusiones

La obtención de estas conclusiones se basa en los objetivos establecidos al principio del documento en el *Capítulo 1: Introducción* y en los resultados obtenidos mediante la experimentación realizada y analizada en el capítulo 4.

Las conclusiones extraídas con el proyecto realizado son las siguientes:

- En ocasiones es conveniente la transformación del espacio de datos para mejorar la precisión de la clasificación. La clasificación en el espacio original (equivalente a la transformación con la matriz identidad) con el algoritmo base utilizado en este trabajo (K-NN) puede ser mejorada con transformaciones adecuadas.

- En los experimentos realizados puede observarse que en todos los dominios es posible igualar o mejorar los resultados obtenidos con la matriz identidad.

- Los resultados obtenidos con los dominios sintéticos rectas ruido horizontales y rectas ruido 45 muestran claramente que el algoritmo funciona correctamente pues encuentra las transformaciones adecuadas para obtener precisiones muy altas, siendo estas prácticamente nulas en el espacio original.

- La matriz que mejores resultados ha obtenido ha sido la matriz completa. Es lógico pensar que este tipo de matriz haya sido la que mejor ha funcionado, ya que es la que permite transformaciones más complejas.

- La matriz que peores resultados ha obtenido ha sido la simétrica. Este tipo de matriz no ha conseguido destacar en ninguno de los dominios experimentados. Por el contrario sí que se ha mostrado más regular que la matriz diagonal, ya que si se observan los dominios sintéticos, se observa que los resultados obtenidos por la matriz simétrica son mejores.

- Se ha obtenido mejor resultado con la función de evolución normal. Este tipo de ejecución permite evolucionar más lentamente la matriz, pero ajustando de una mejor manera los valores de sus elementos. Este método de evolución no provoca grandes “saltos” en el conjunto de soluciones.

- El objetivo del proyecto por tanto ha sido alcanzado, ya que se ha desarrollado un algoritmo capaz de evolucionar matrices de transformación para logar un mayor porcentaje de acierto a la hora de clasificar nuevos datos.

5.2 Líneas futuras

A continuación se describen algunas posibles líneas futuras de investigación:

- **Uso de otros algoritmos para obtener el fitness de los individuos**. En este proyecto el algoritmo utilizado para obtener el fitness de los diferentes individuos es K-NN (ver *Capítulo 2* para más detalle).

  Una posible línea de investigación sería extender el método para poder utilizar diferentes algoritmos de clasificación.

- **Obtención de descendientes**. Durante el desarrollo de este proyecto se han implementado dos métodos (normal y N hijos) diferentes para obtener descendientes a partir del individuo principal.

  Se plantea la posibilidad de incorporar nuevos métodos para la obtención de descendientes. Se plantea como posible mejora un método mixto que utilice lo dos desarrollados, es decir, que genere un número de hijos igual al número de elementos de la matriz que compone al individuo (caso de evolución normal) y que cada uno de esos hijos pudiera tener alterados cualesquiera de sus elementos de acuerdo a una probabilidad (caso N hijos).

- **Evolución de los individuos**. El proyecto desarrollado incluye un método de evolución de individuos basado en la variación o mutación de sus elementos mediante el uso de una distribución normal.

  Una futura línea de investigación podría usar la técnica de algoritmos genéticos, en los que se dispone de un conjunto de individuos iniciales a partir de los cuales se comienza a evolucionar. En el proyecto realizado se parte de un único individuo y por lo tanto no se realiza reproducción entre individuos, que sí podría realizarse en el caso de los algoritmos genéticos.

Anexo I

Manual de Usuario

Importar el proyecto en eclipse

En este apartado del manual de usuario se detallará la forma en la que el usuario podrá cargar el proyecto en el entorno de desarrollo Eclipse.

El primer paso es lanzar el programa eclipse y seleccionar el espacio de trabajo en el que se va a trabajar (workspace). Una vez seleccionado el *workspace* pulsar sobre el botón *OK*.

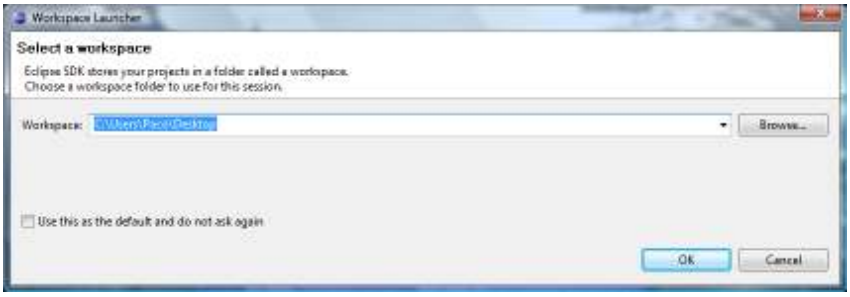


Imagen 66: Selección de workspace

Una vez elegido el workspace se inicia el programa Eclipse. Tras haber iniciado el programa Eclipse es necesario importar el proyecto que contiene el algoritmo de búsqueda local implementado. Para ello, acceder mediante *File* →  .

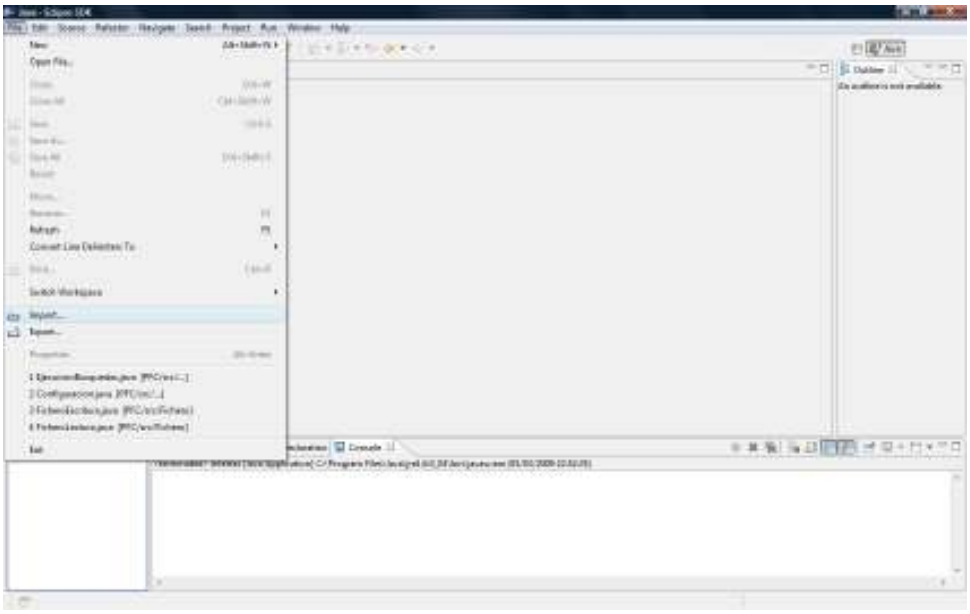


Imagen 67: Importar un proyecto

Una vez se ha accedido a la opción de *Import*, se abrirá una nueva ventana que permitirá seleccionar el tipo de proyecto que se quiere importar. En el caso de este proyecto, se seleccionará la opción *General* → *Existing Projects into Workspace*.

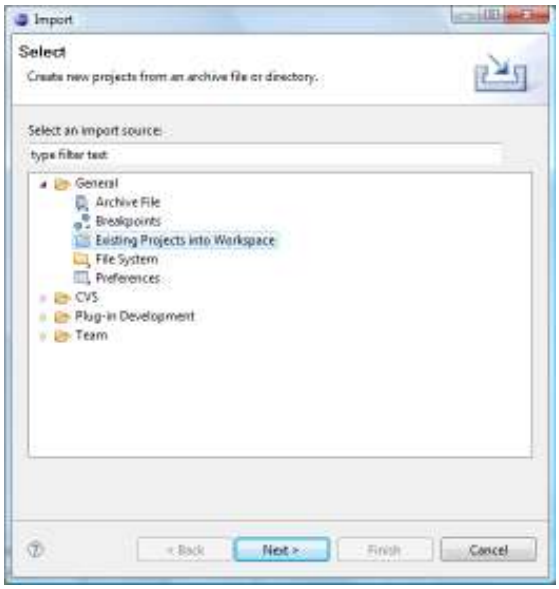


Imagen 68: Tipos de proyectos a importar

Seleccionado el tipo de proyecto a importar, se pulsará en el botón *Next*. Al pulsar, se mostrará una nueva ventana que nos permitirá buscar proyectos, usando el botón *Browse*. Cuando un directorio haya sido elegido, se mostrará en la parte principal de la ventana (Projects) los posibles proyectos que se pueden importar. En este momento se selecciona el proyecto PFC y se pulsará el botón *Finish*.

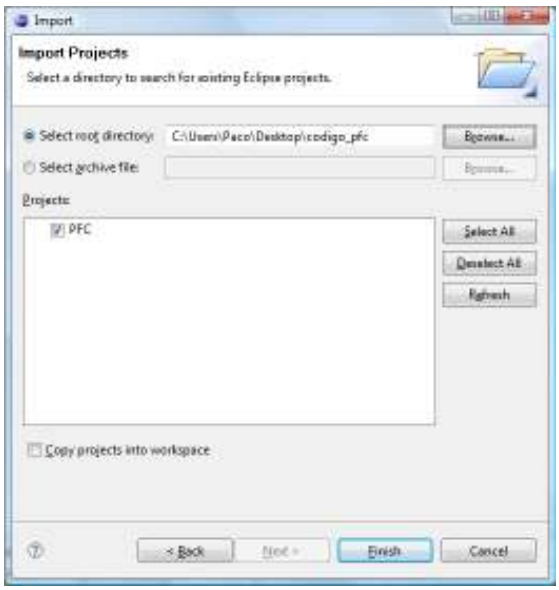


Imagen 69: Posibles proyectos para ser importados

Una vez finalizado estos pasos, el proyecto se encontrará cargado en el eclipse. Se puede ver la estructura del proyecto en la parte izquierda de la pantalla.

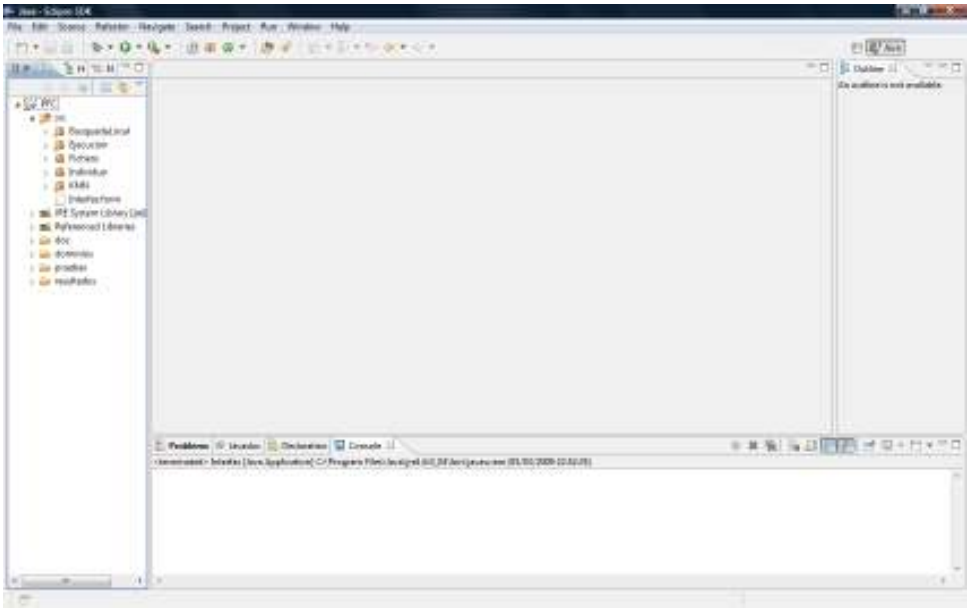


Imagen 70: Proyecto importado

Para finalizar la carga del proyecto y estar en disposición de poder ejecutar el proyecto es necesario además cargar dos librerías. Estas librerías son las siguientes, y se encuentran almacenadas en el mismo directorio que contiene el proyecto eclipse.

- **Jama-1.0.2.jar** → Librería para usar matrices.
- **psimj.jar** → Librería para utilizar distribuciones Normales.

Para añadir estas dos librerías es necesario pulsar el botón derecho sobre el proyecto, y posteriormente sobre la opción *Properties*, tal y como se observa en la siguiente imagen.

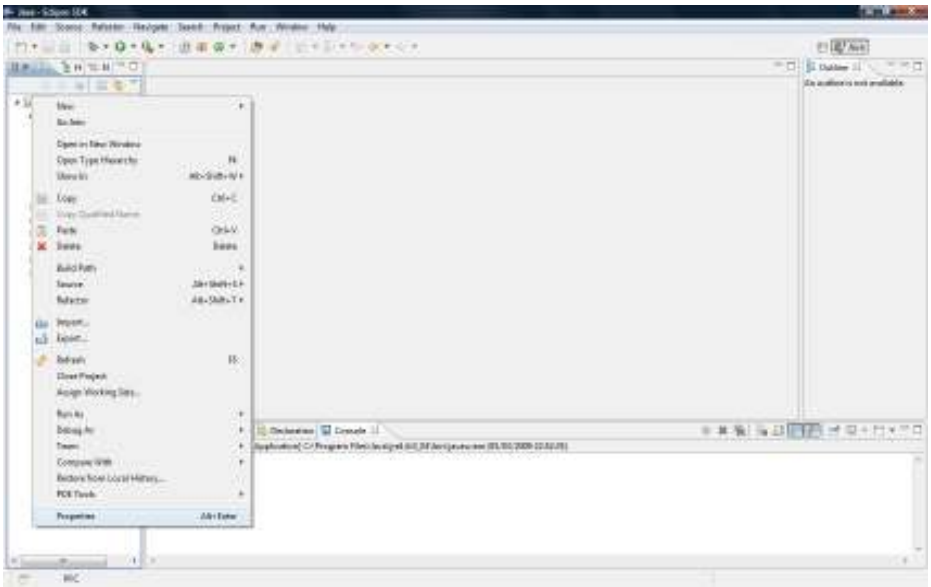


Imagen 71: Seleccionando la opción *Properties*

Al seleccionar esta opción, se abrirá una la ventana en la que se seleccionará *Java Build Path* en el menú de la izquierda. Tras esto es necesario seleccionar la pestaña Libraries en la parte central de la ventana.

Utilizando el botón *Add External JARs* se añadirán las librerías que se comentaron anteriormente. Finalmente se pulsará el botón *OK*.

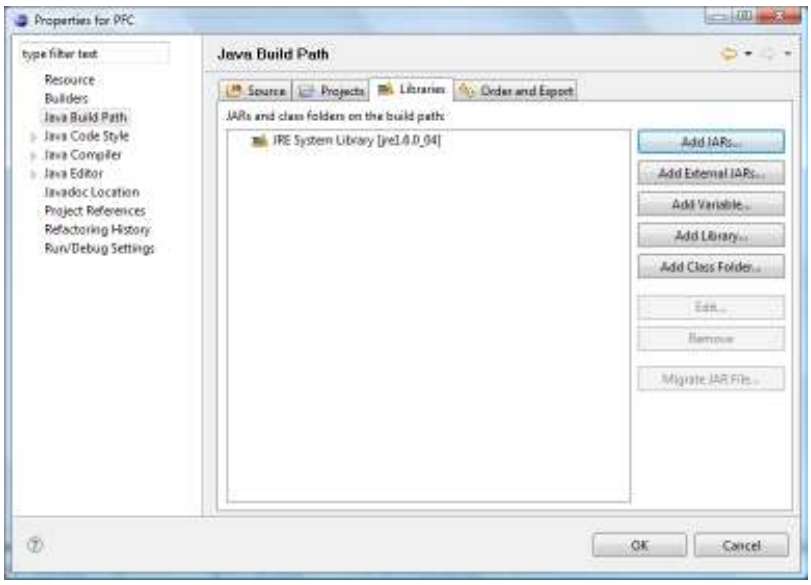


Imagen 72: Ventana *Properties*

Una vez finalizado este proceso el proyecto se encuentra totalmente cargado en el entrono y preparado para ser ejecutado.

Ejecutar la aplicación en Eclipse

Los siguientes apartados describirán la aplicación y ayudarán al usuario a conocer los aspectos claves para la configuración de la misma antes de la ejecución.

Ejecución normal

La ejecución del programa implementado para realizar búsqueda local en los diferentes dominios que el usuario desee, requiere que se hayan realizado los pasos descritos en la sección ***Cargar el proyecto en Eclipse***.

El usuario deberá pulsar sobre el botón  . De esta forma el programa comenzará su ejecución.

Primera ejecución

La primera ejecución que se realiza tras importar el proyecto mostrará una serie de ventanas que solamente se mostrarán en esta primera ejecución (*nótese que una vez importado el proyecto, la próxima vez que el usuario vuelva a abrir Eclipse con el mismo workspace éste se cargará automáticamente*).

La forma de iniciar esta primera ejecución es similar a la descrita en el apartado ***Ejecución normal***, es decir, pulsando el botón  situado en la barra de herramientas de la parte superior de la pantalla.

La primera ventana que el usuario se encontrará será la que le permita elegir la forma de ejecutar el programa. En esta ventana, el usuario seleccionará la opción *Java Application.*

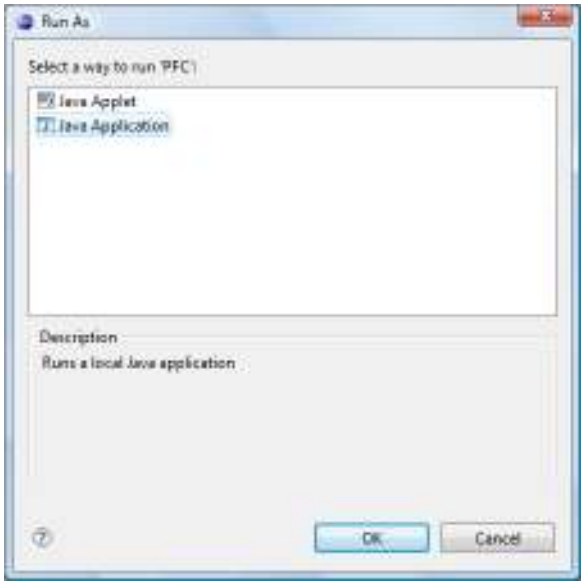


Imagen 73: Forma de ejecución

Una vez seleccionada la forma de ejecución, se selecciona la clase que contiene el **main** y que por tanto es el punto de inicio de la ejecución.

En esta ventana, tal y como se muestra a continuación, seleccionar la clase *Interfaz* y pulsar sobre el botón *OK*.

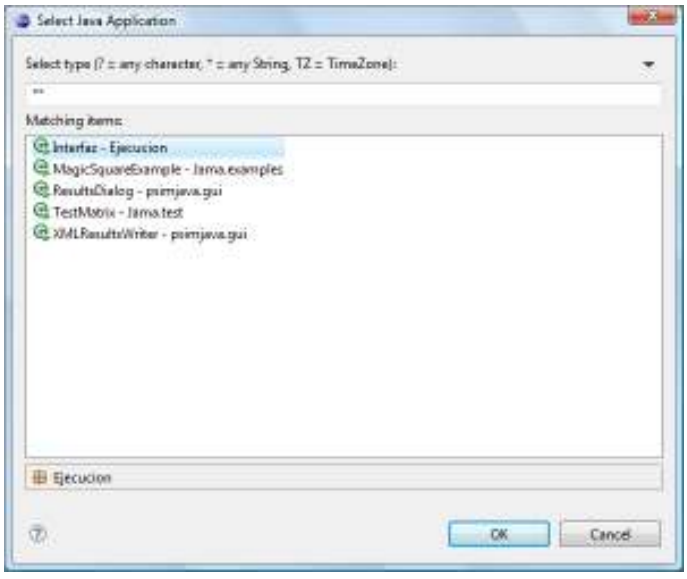


Imagen 74: Seleccionar clase principal

Pulsando *OK*, el programa comienza su ejecución.

Uso de la aplicación

La aplicación dispone una ventana de principal en la que el usuario podrá configurar y ejecutar el programa con los parámetros deseados.

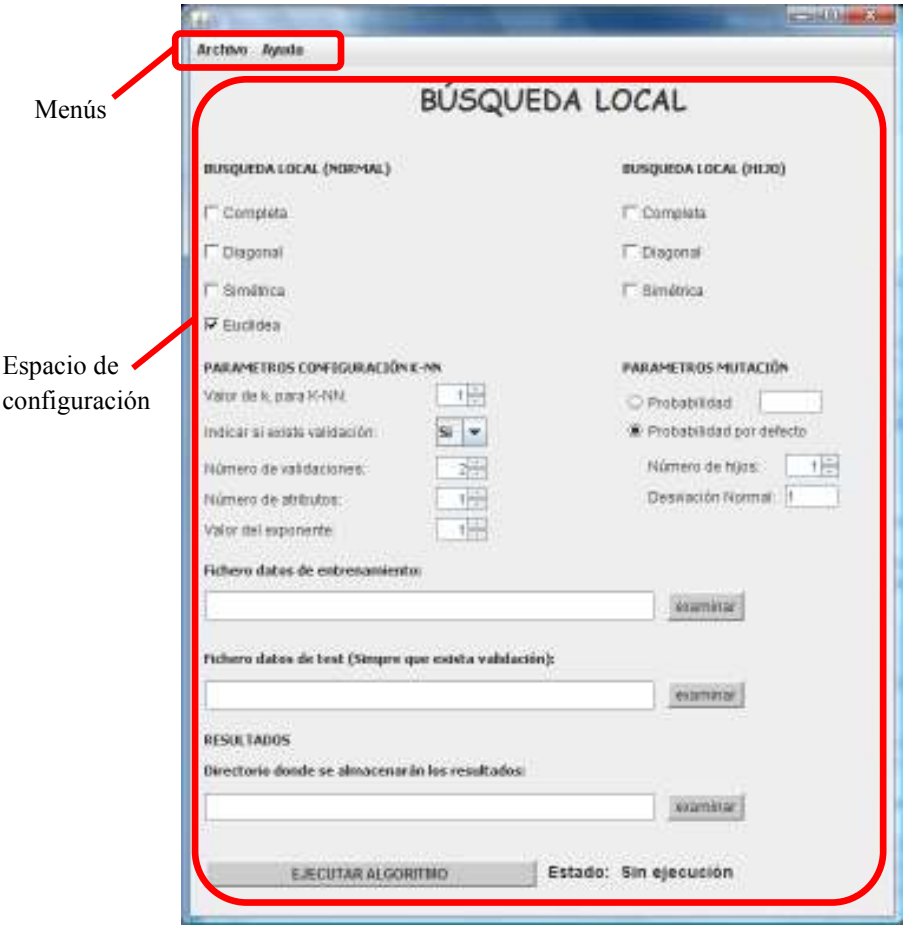


Imagen 75: Ventana principal de la aplicación

Los siguientes apartados detallarán cada una de las diferentes opciones que el usuario podrá configurar para ejecutar el programa.

Menú Archivo

La vista del menú *Archivo* es la siguiente:

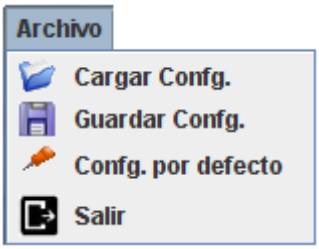


Imagen 76: Menú *Archivo*

A continuación se describen las distintas opciones o funcionalidades que aporta este menú.

- **Cargar Confg**. Esta opción permitirá al usuario seleccionar y cargar un fichero de configuración que previamente haya guardado.
- **Guardar Confg.** Esta opción almacena en un fichero indicado por el usuario la configuración que esté actualmente en la ventana principal.
- **Confg. por defecto**. Mediante esta opción el usuario restablecerá la configuración establecida por defecto.
- **Salir**. Esta opción permitirá al usuario cerrar la aplicación y salir de la misma.

Menú Ayuda

La vista del menú *Archivo* es la siguiente:

Imagen 77: Menú *Ayuda*

La opción **Acercar de Búsqueda Local** abrirá una nueva ventana con información referente al programa, tal como autor, tutores y versión.

Espacio de configuración

La aplicación comenzará su ejecución mostrando una ventana que permitirá al usuario realizar una configuración de las diferentes opciones que permite el programa.

El usuario podrá establecer el tipo de búsqueda que desea realizar sobre los datos. Dispone de dos tipos:

- **Búsqueda normal**. La mutación de un individuo genera un número de hijo igual a la dimensión del individuo al cuadrado. Por ejemplo, si el individuo es de dimensión tres, la mutación generará nueve hijos.
- **Búsqueda con hijos**. La mutación de un individuo genera un nuevo hijo. Si no mejora el fitness del padre, generará otra y así hasta alcanzar el máximo número de hijos establecido por el usuario en la opción correspondiente.

Las opciones que contemplan la configuración del algoritmo K-NN en esta ventana son las siguientes:

- **Valor de k para K-NN**. El usuario indicará en esta opción el valor del parámetro *k* para el algoritmo K-NN que utiliza la búsqueda local para entrenar.
- **Validación**. En este apartado el usuario indicará si se realizarán validaciones o no.
- **Número de validaciones**. Opción que permitirá al usuario establecer el número de validaciones que se realizarán, siempre y cuando en la opción anterior se haya indicado *Si*.
- **Número de atributos**. El usuario debe indicar en este punto el número de atributos que contienen los datos pertenecientes al dominio que el usuario vaya a introducir.
- **Valor del exponente**. El usuario puede desear realizar una búsqueda lineal, para lo cual el valor de exponente será 1; o por el contrario puede desear realizar una búsqueda en **n** dimensiones, para lo cual pondrá el valor deseado en esta opción.
- **Fichero de datos de entrenamiento**. El usuario introducirá en esta opción el fichero que contiene los datos del dominio que desea utilizar para realizar el entrenamiento. Si se usan validaciones, se utilizarán los datos del fichero seleccionado en este punto.
- **Fichero de datos de test**. Si el usuario **no** desea realizar validación, es obligatorio introducir un fichero en esta opción. Este fichero se utilizará para realizar *test* del resultado obtenido con el conjunto de datos del fichero de entrenamiento.

La configuración de los parámetros que gestionarán la mutación de los individuos se encuentra también en la ventana principal, en el apartado *PARAMETROS MUTACIÓN*. Estos parámetros son los siguientes:

- **Probabilidad**. El usuario podrá indicar la probabilidad de mutar un elemento del individuo para el caso hijo, o por el contrario seleccionar la probabilidad por defecto implementada en el programa. El valor introducido para indicar la probabilidad debe ser un valor entre 0.0 y 1.0.

- **Número de hijos**. En esta opción el usuario podrá indicar el número de hijos máximo que el algoritmo de búsqueda local para el caso con N hijos podrá tener.

- **Desviación de la Normal**. Para realizar la mutación de los elementos de un individuo se emplea una distribución Normal, $N(\mu, \sigma)$. Para el caso concreto del algoritmo se utiliza un valor $\mu = 0$ (media) y se deja a elección del usuario el valor de σ (desviación).

En la parte inferior de este espacio de configuración, el usuario configurará las opciones necesarias para obtener los datos de entrada al algoritmo y para almacenar los datos de salida generados por el programa.

- **Fichero datos de entrenamiento**. El usuario indicará en este parámetro la ruta en la que se encuentra el fichero de datos que contiene los datos que usará el programa para realizar la fase de entrenamiento.

- **Fichero datos de test**. Esta opción será de carácter obligatorio siempre y cuando el usuario haya establecido que no va a realizar validación. Se indicará la ruta del fichero que usará el programa para realizar la fase de test.

- **Directorio donde se almacenarán los resultados**. En este punto se indicará el directorio donde el usuario desea almacenar los resultados generados por el programa (Para más detalle ver apartado *Directorio datos de salida*).

Por último, una vez el usuario ha establecido las opciones deseadas, pulsar sobre el botón *EJECUTAR ALGORITMO* para comenzar las búsquedas locales seleccionadas.

Directorio datos de salida

El usuario indicará el directorio en el que desea almacenar los datos generados por el programa durante su ejecución.

La ejecución del programa genera una serie de directorios y de ficheros de salida que son almacenados en un esquema de directorio. A continuación se presenta un ejemplo en el que el usuario indicó que los datos de salida generados se almacenaran en el directorio *resultados*.

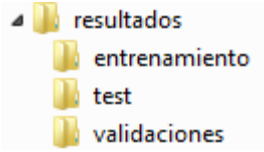


Imagen 78: Esquema de directorios de salida

Los ficheros generados por el programa se almacenarán en el directorio *resultados*. Para ello, en el directorio especificado se generarán otros tres directorios que contendrán la siguiente información:

- **Directorio entrenamiento**. Contendrá ficheros con líneas que permitirán visualizar la evolución del fitness de entrenamiento y de la matriz de pesos para cada una de las validaciones (siempre que haya validaciones, o para el conjunto de entrenamiento sino hay validación).
- **Directorio test**. Contendrá ficheros que reflejarán los individuos finales obtenidos durante el entrenamiento mostrando el fitness de test, de entrenamiento y los valores de la matriz de pesos.
- **Directorio validaciones**. Contendrá las diferentes validaciones que se han generado a partir del fichero de datos del dominio.

Anexo II

Bibliografía

[1] *Local search in combinatorial optimization*; Emile Aarts & Karel Lenstra; Ed. Princenton University Press, 2003

[2] *Stochastic Local search Foundation and Applications*; Holger H. Hoos & Thomas Stützle; Ed. Morgan Kaufmann, 2005

[3] *Machine Learning*; Tom Michael Mitchell; Ed. McGraw Hill, 1997

[4] *An introduction to genetic algorithms*; Melanie Mitchell; Ed. MIT Press, 1996

[5] *Machine learning and algorithmic perspective*; Stephen Marsland; Ed. Chapman & Hall / CRC, 2009

[6] *Locally Weighted Learning*; Christopher G. Atkeson, Andrew W. Moorey and Stefan Schaalz; Revista: Artificial Intelligence Review; Ed. Springer Netherlands, 1997; volumen 11; página 12.

[7] *Eclipse*; Jim D'Anjou, Sherry Shavor, Scott Fairbrother, Dan Kehn, John Kellerman, Pat McCarthy; Ed. Addison-Wesley, 2003